JLO SAFE V2.65 Disassembly

4/96

This file contains the source code for JLO SAFE V2.65, its object code, and an assembler to assemble the object code from the source code.

I, John Oliger, still maintain copyrights of this code, but am releasing this as 'freeware' to give Oliger Disk System customers access to the code for their own modifications or education. You may modify this code, as you desire, but any such code released for others to use must be labeled that it WAS so modified and as such is not supported by John Oliger Co. You must also maintain my copyright by its declaration along with your own.

Also in this archive is a freeware assembler written by a friend of mine by the name of Dave Gibbons. Dave gave me permission to release this assembler with my source code, with the stipulation that his copyright be maintained and that no support of the program itself is offered. It may interest you to note that this assembler will also assemble 6502 (he calls it 8502 as that was the chip used in the Commodore 128 computer) and the Signetics 87C751 single chip microcontroller. You can use the SAFE source code as an example of the syntax of the assembler.

JLO SAFE was originally written using Ray Kingsley's "Hot Z" assembler, but was later converted over to this assembler for ease of modification.

You may upload this archive to your favorite BBS or FTP site as long as the archive is intact with all files including this readme file.

Thanks,

John L. Oliger
joliger@in.net
4/9/96

—

```
>>PON
>MCOL !14
>CCOL !34
>>LABEL A,!10,!165, ;This one
>>LABEL A,!12,!167
;JLO SAFE V2.65
;(C)1985-1993, John L. Oliger
;All Rights Reserved

;Often changed SAFE variables
EQU DFTRACKS=!40 ;Normal default tracks=40d
;;EQU DFTRACKS=!80 ;Tracks=80d
EQU DFSIDES=2  ;Normal default max sides is two
EQU DFSTEP=0   ;Normal step rate is fastest
EQU LNFEED=0   ;Normal default is a null after a CR
```

```
EQU COPYFLAG=0 ;Default to 0 for ASCII COPY/
               ;(0=ASCII, 1=Okidata, 2=Olliv, 3=Gemini, 4=GB)

;SAFE variables not usually changed
EQU DTKS=2600  ;# of tracks on disk (CAT)
EQU DSDS=2601  ;# of sides on disk (CAT)
EQU MXCY=2602  ;Total cylinders on disk (CAT)(2X)
EQU FRCY=2604  ;Free cylinders on disk (CAT)(2X)
EQU NXCY=2606  ;Next avail free cyl track# (CAT)
EQU NXSD=2607  ;Side# of next free cyl (CAT)
EQU NXCA=2608  ;Address of next free CAT entry (CAT)(2X)
EQU DNAM=2610  ;Disk name area. 16 characters
EQU CTFL=2620  ;Catalog file area start
EQU CBUF=3400  ;Cat buffer area (erase)
EQU TCAT=3720  ;Temp CAT build-up area (3720-3735)
EQU TTYP=372A  ;3720-3729=name, 372A=type
EQU TLEN=372B  ;Length (2X)
EQU TBEG=372D  ;Start (2X)
EQU TOFS=372F  ;Offset (2X)
EQU TTRK=3731  ;Track (1X)
EQU TSID=3732  ;Side (1X)
EQU TCYL=3733  ;Cylinders (1X)
EQU FLAD=3734  ;Add of CAT entry matching entry in TCAT(2X)
EQU OPGM=3778  ;Address of current Basic prog (Merge)(2X)
EQU WFLG=377A  ;SAFE warning message flag (1X)
EQU PFLG=377B  ;Decimal # print flag (1X)
EQU LPCN=377E  ;Max loop count (For/Next) (2X)
EQU DNUM=3780  ;Current drive # (1X)
EQU TRKS=3781  ;Max # of tracks per side (1X)
EQU SIDS=3782  ;# of sides (1 or 2) (1X)
EQU DSEL=3783  ;Soft copy of port B7H (1X)
EQU GSFLAG=3784 ;Gosub/ flag. 0=not loaded or <>0=loaded (1X)
EQU SIZE=3785  ;# of cylinders reclaimed (Erase)(1X_)
EQU HDSP=3786  ;Drive headstep speed (1X)
EQU SID#=3787  ;Current side # (1X)
EQU TRK#=3788  ;Current track # (1X)
EQU SEC#=3789  ;Current sector # (1X)
EQU T/SP=378A  ;Timex/Spectrum rom flag (1X)
EQU COVR=378B  ;Copy over flag (1X)
EQU COPF=378C  ;Copy/ flag (1X)
EQU SV/L=378D  ;Save/Load/Merge/RUN flag
;(0=SAVE, 80=MERGE, F7=RUN, FF=LOAD) (1X)
EQU FORF=378E  ;FOR/ flag (1X)
EQU WIDECAT=378F ;CAT type flag. 0=norm or <>0=# of columns
;in V2.6 & up wide CAT (CAT N)(1X)
EQU OLHL=3790  ;Temp store for HL (2X)
EQU OLDE=3792  ;Temp store for DE (2X)
EQU TEMPA=3794 ;Temp store for reg A (1X)
EQU SYNSAVE=3795 ;Temp store for Basic variable flgs (1X)
```

```
EQU TRYS=3797  ;Retry counter (1X)
EQU TRY2=3798  ;Alternate retry counter (1X)
EQU NXC'=3799  ;Address of next char to interpret (NEXT)(2X)
EQU CPP'=379B  ;Looping line # (NEXT)(2X)
EQU NPA'=379D  ;Looping line address (NEXT)(2X)
EQU CPS'=379F  ;Looping line statement # (NEXT)(2X)
EQU INTF=3EE8  ;Interrupt flag (1X)
EQU I_ST=3EE9  ;Temp store for I reg (1X)
EQU IYST=3EEA  ;Temp store for IY (2X)
EQU IXST=3EEC  ;Temp store for IX (2X)
EQU BC'S=3EEE  ;Temp store for BC' (2X)
EQU DE'S=3EF0  ;Temp store for DE' (2X)
EQU AF'S=3EF2  ;Temp store for AF' (2X)
EQU AFST=3EF4  ;Temp store for AF (2X)
EQU BCST=3EF6  ;Temp store for BC (2X)
EQU DEST=3EF8  ;Temp store for DE (2X)
EQU HLST=3EFA  ;Temp store for HL (2X)
EQU HL'S=3EFC  ;Temp store for HL' (2X)
EQU SPST=3EFE  ;Temp store for SP (2X)
EQU SRSD=3F00  ;Src drive side # (MOVE)(1X)
EQU DESD=3F01  ;Dest drive side # (MOVE)(1X)
EQU EADR=3F02  ;Pointer within CAT buffer, used by ERASE(2X)
EQU NMIF=3F0B  ;NMI SAVE signal flag (1X)
EQU FFST=3F0D  ;Video port status store (1X)
EQU MODF=3F0E  ;Interrupt mode flag (1X)
EQU R_ST=3F0F  ;Temp store for R reg (1X)
EQU DSL'=3F10  ;DSEL of src drive (MOVE) (1X)
EQU DSL2=3F11  ;DSEL of dest drive (MOVE) (1X)
EQU STK#=3F12  ;Src drive current track# (MOVE)(1X)
EQU DTK#=3F13  ;Dest drive current track# (MOVE)(1X)
EQU SSDS=3F14  ;Src drive max sides (MOVE)(1X)
EQU DSTS=3F15  ;Dest drive max sides (MOVE)(1X)
EQU TCNT=3F16  ;Counter used in MOVE commands (1X)
EQU SDV#=3F17  ;Src drive# (MOVE)(1X)
EQU DDV#=3F18  ;Dest drive# (MOVE)(1X)
EQU CODF=3F1B  ;Flag used for BYTE or DATA save/load (1X)
EQU STOR=3F30  ;Temp store area used by bankswitch routine(16X)
EQU STAF=3FFD  ;Temp store area for regs AF (2X)
EQU STHL=3FFE  ;Another store location for HL (2X)
EQU chas=5C36  ;Basic rom's pointer to character dot table
EQU ernr=5C3A  ;IY=5C3A  Error # -1
EQU flgs=5C3B  ;Basic flags
EQU ersp=5C3D  ;Error stack pointer
EQU nwpc=5C42  ;Basic next line address
EQU nspc=5C44  ;Basic next statement address
EQU ppc_=5C45  ;Basic program counter
EQU sbpc=5C47  ;Basic statement counter
EQU vars=5C4B  ;Basic variables pointer
EQU chan=5C4F  ;Basic's current output channel
```

```
EQU prog=5C53  ;Basic program pointer
EQU nxln=5C55  ;Basic's pointer to start of next line
EQU elin=5C59  ;Edit line addr
EQU chad=5C5D  ;Basic's pointer to current interpret char
EQU xptr=5C5F  ;Basic error pointer
EQU stnd=5C65  ;Basic stack end pointer
EQU ppos=5C7F  ;Current print pos
EQU mem5=5CAB  ;Basic calculator scratchpad
EQU rmtp=5CB2  ;T/S sysv RAMTOP
EQU errt=5CBB  ;Timex ON ERR flag
EQU vmod=5CC2  ;Basic sv showing current video mode

ORG 0000
RESET DI         ;No interrupts
     XOR A       ;A=00
     OUT (F4),A  ;Dock & Exrom off
     JP STSV     ;Cont later
     NOP         ;Pad till RST08H

;RST 08H=Disk B entry point
ESTR DI          ;No int
     RST 28H     ;Cont at 0028H
B_ON DI          ;No int
     RET         ;Bank on & ret
     RST 18H     ;RST18H will ret to Home (dock) bank
     RST 18H
     RST 18H
     RST 18H

;RST 10H=Advance to next valid Basic character (Basic's RST20H)
RST10 LD (STHL),HL ;Save HL
      LD HL,0020 ;Call 0020 in Basic's rom
      JR RST20A  ;Jump to continue

;RST 18H=Jump to (SP) in another bank
RST18 INC SP     ;Trash ret addr
      INC SP
      PUSH AF    ;Save AF
      JP CALL    ;Jump to (SP) in rom
      DB 0,0     ;Pad till RST 20H

;RST 20H=Call (SP) in another bank
RST20 LD (STHL),HL ;Save HL
      POP HL     ;Get ret addr
      EX (SP),HL ;Exchange w/dest addr on stack
RST20A PUSH HL   ;Restack dest addr
      JR CONT20  ;Continue later

;RST 28H either continues RST 08H (if ret address is from page
```

```
;zero) or accesses SAFE's function dispatcher. If the dispat-
;cher is accessed, a data byte following the RST 28H will spec-
;ify which function is desired. (See dispatcher data sheet)
RST28 EX (SP),HL ;Get return address
      INC H      ;Is return address from page zero?
      JP CONT28  ;Continue later
BUSY? RLA        ;Controller busy?
      JR NC,ERROR ;Error if not busy

;RST 30H=Send byte in D to controller port C
SEND: IN A,(8F)  ;Get controller status (this is RST 30H)
      RRA        ;Ready for another byte?
      RRA
      JR NC,BUSY? ;Loop if not
      OUT (C),D  ;Send the byte

;RST 38H & Mode 1 int handler
RST38 RET

;Here if controller not busy; ERROR!
ERROR POP AF     ;Clear ret addr from stack
      SCF        ;Sig an error
      RET        ;Ret one level (call) back

;Here to evaluate the current expression as an integer
;Will return to rom in syntax time
EVA#: CALL NXEX  ;Evaluate expression
      JP NC,ER_C ;Error C if a string
      CALL SYRT  ;Ret to rom in syntax time
      RET        ;Else ret to SAFE command

;Xor (HL) in other bank with A. Ret result in A
XRHL: PUSH HL    ;Save HL
      CALL GTHL  ;Get (HL) in other bank into L
      XOR L      ;Xor contents with A
      POP HL     ;Restore HL
      RET

ORG 004D
GOHM: LD A,(0008) ;B bank off

;Insure next char is "=" and advance to char after that
AFTEREQ: RST 10H  ;Advance to next Basic char
      CP '='      ;"="?
      JP NZ,ER_C  ;Error C if not
      RST 10H     ;Advance to char after "="
      RET

;Here to send command in reg A to disk controller & allow
```

```
;a slight pause giving it time to digest command
SEND8F OUT (8F),A ;Send command to controller
      LD B,07     ;Initial count
SEND8FX DJNZ SEND8FX ;Pause to allow controller time
      RET

      ORG 0060   ;Start at 0060
      DB 0,0,0,0,0,0 ;0060-0065 are unused for Larken compat.

;This code is at 66H & is the NMI handler
;Code from 0068-006E must stay NOP as MI must be active
;on each fetch between 0068 and 006F
NMIH: PUSH AF     ;Duplicate code in rom, 0066 here
      PUSH HL     ;Ditto
      DB 0,0,0,0,0,0,0
      JP NMSA     ;Go handle NMI

CONT20 LD HL,000A
      EX (SP),HL  ;000A on stack under dest addr
      PUSH HL     ;Dest of CALL back onto stack
      PUSH AF     ;Save AF. Will be restored in the other bank
      LD HL,(STHL) ;Restore HL
CALL: IN A,(F4)   ;Get dock chunk sel status
      RRA         ;Chunk 0 on?
      JR NC,GOHM  ;Go to home bank if not
      IN A,(FF)   ;Get vid port stat
      RLA         ;Check exrom sel bit
      JR NC,GOHM  ;Jump to home if exrom off
      JP GOEX     ;Go to exrom

SAFEFUNC LD (TEMPA),A ;Save A
      LD A,(HL)   ;Get function data byte
      INC HL      ;Point past data byte
      EX (SP),HL  ;Store updated ret address and restore HL
      CP 2B       ;Function implemented yet?
      JR NC,FUNCDONE ;Do nothing & ret if not defined
      PUSH HL     ;Save HL
      LD HL,FUNCTABLE ;Point at function address table
      ADD A,A     ;Double value of function code for look-up
      ADD A,L     ;Add offset to start address
      LD L,A
      JR NC,FUNC2 ;Skip if no overflow
      INC H       ;Wrap to next page
FUNC2 LD A,(HL)   ;Get LSB of function address
      INC HL      ;Bump
      LD H,(HL)   ;Get MSB of function address
      LD L,A      ;Form full pointer
      EX (SP),HL  ;Funtion address to stack & restore HL
FUNCDONE LD A,(TEMPA) ;Restore A
```

```
      RET         ;Goto either function or calling routine

CONT28 DEC H      ;Finish page zero test
      JR NZ,SAFEFUNC ;Safe fuction dispatch if not page zero
      POP HL      ;Correct stack for no ret address needed
      LD HL,(chad) ;Get Basic char addr
      DEC HL      ;Point at char before error
      LD A,(HL)   ;Get it
      CP F3       ;NEXT?
      JP Z,NEXT
      LD B,A      ;Save char
      LD (SPST),SP ;Save stack pos
      POP HL      ;Get error# addr
      PUSH HL
      CALL GTHL   ;Get error# in L
      XOR A       ;A=0
      LD (SV/L),A ;Reset SAVE/LOAD/MERGE sysv
      LD A,L      ;Get error#
      CP 0B       ;Error C?
      JR Z,CONT8  ;OK if error C
      CP 17       ;Error 18?
      JR NZ,REER  ;Error if neither
CONT8 CALL OFEX   ;Insure exrom off & reset bs SP
      LD HL,(chad) ;Get current char add
      LD A,(HL)   ;Get char
      CP '/'      ;Is it "/"?
      JR Z,HAND_/
      LD A,B      ;Get back char before present one
      CP CF       ;CAT?
      JP Z,HCAT
      CP EF       ;LOAD?
      JP Z,HLD0
REER: LD HL,(chad) ;It's a real error so get char addr
      LD (xptr),HL ;Point Basic error pointer to it
      POP HL      ;Get addr where err occured
      CALL GTHL   ;Err code to reg L
ENTL  CALL OFEX   ;Insure exrom is off & reset bs SP
      LD DE,0055  ;0055=rom error handler addr
      LD SP,(SPST) ;Restore stack addr
      PUSH DE     ;0055 on stack
      RST 18H     ;Jump to rom addr 0055

;Handle the commands with "/" in them
HAND_/ DEC HL     ;Point at char before "/"
      LD A,(HL)   ;Get it
      INC HL      ;Point back at current char
      CP F8       ;SAVE?
ZSAVE JP Z,H/SA
      CP DF       ;OUT?
```

```
      JR Z,ZSAVE  ;Save w/OUT command,too
      CP EF       ;LOAD?
ZLOAD JP Z,H/LD
      CP BF       ;IN?
      JR Z,ZLOAD  ;Load w/IN command,too
      CP F7       ;RUN?
      JR NZ,HND/2 ;Skip if not RUN/
      LD A,(flgs) ;Get Basic's Syntax flag
      LD (SYNSAVE),A ;Save Basic's flag
      SET 7,A     ;Force run-time mode
      LD (flgs),A
      LD A,F7     ;RUN token back to A
      JP SVCN     ;Continue RUN in common code
HND/2 CP ED       ;GOSUB?
      JR Z,H/GOSUB
      CP D5       ;MERGE?
      JP Z,H/MG
      CP FF       ;COPY?
      JP Z,CPY/
      CP F1       ;LET?
      JP Z,LET/
      CP D1       ;MOVE?
      JP Z,MV/H
      CP D6       ;VERIFY?
      JP Z,VF/H
      CP D0       ;FORMAT?
      JR Z,FRM/
      CP D2       ;ERASE?
      JP Z,H/ER
      CP E5       ;RESTORE?
      JP Z,RX/X
      CP CF       ;CAT?
      JP Z,CAT/
      CP EB       ;FOR?
      JP Z,FOR/
      JR REER     ;None of above, so a real error

;GOSUB/ handler
H/GOSUB CALL SYN? ;Syntax time?
      JR NC,GOSUB1 ;Skip to end in syntax time
      LD A,(GSFLAG) ;Get GOSUB/ flag
      AND A       ;Zero?
      JP Z,ER_S   ;File not found report if not loaded
GOSUB1 CALL 3800  ;Call previously loaded mc
      CALL ENDLINE ;Skip all characters till end of line
      JP DONEOK   ;Return via common code

ENDLINE LD DE,0018 ;RST18H is Basic's get current char routine
      PUSH DE     ;Address to stack
```

```
      RST 20H     ;Call routine
END1  CP 0D       ;CR?
      RET Z       ;Done if CR ends statement
      CP ':'      ;":"?
      RET Z       ;Done if ":" ends statement
      RST 10H     ;Else advance to next character
      JR END1     ;Skip all characters till end marker found

;Format command handler
FRM/  RST 10H     ;Advance to char after "/"
      CALL NXEX   ;Eval next expression
      JP C,ER_C   ;Error C if numeric
      LD (DEST),DE ;Save pointer to string
      CALL SYRT   ;Ret in syntax time
      LD (BCST),BC ;Save diskname length
      LD A,C      ;Get LSB
      AND A       ;Zero?
      JR NZ,NNUL  ;OK if not null string
ER_F  LD L,0E     ;Ret to Basic with err F
      JP ER_L
NNUL  CALL MFRM   ;Format the disk
      CALL RS02   ;Restore to side 0, track 0
      CALL LCT2   ;Load CAT
      LD HL,(TRKS) ;Get max # of tracks & amount of sides
      LD (DTKS),HL ;Store them in CAT area
      LD A,H      ;# of sides to A
      LD H,00     ;Clear H
      LD B,H      ;Let BC = # of tracks, too
      LD C,L
      DEC A       ;Only 1 side?
      JR Z,X1SID  ;Cyls=tracks if 1 sided
      ADD HL,BC   ;Else cyls=tracks*2
X1SID DEC HL      ;Less 1 cyl used by system
      LD (MXCY),HL ;Store max free cyls
      LD (FRCY),HL ;Free cyls is same on clean disk
      LD HL,0100  ;Assume next cyl=side 1/track 0
      AND A       ;Double sided disk?
      JR NZ,X2SID ;Assumption correct if it is
      DEC H       ;Else next cyl=side 0/track 1
      INC L
X2SID LD (NXCY),HL ;Store pointer to next free cyl
      LD HL,(DEST) ;Retrieve pointer to disk name
      LD A,(BCST) ;& length of diskname
      CALL MDNA   ;Pad or truncate name as req & move it
      EX DE,HL    ;Point at first file entry location (CTFL)
      LD (HL),80  ;First file entry is end of CATalog
      LD (NXCA),HL ;Also next CAT entry location
      CALL SCT2   ;Save the new CATalog
      POP HL      ;Prepare stack for re-entry into Basic
```

```
      CALL CAT2   ;Perform an automatic CATalog
      JP RETB     ;Ret to Basic

;Prepare exchange registers for a LOAD
;On entry, DE=number of bytes to load
;On exit, DE'=# of bytes to pad out a .5K cyl for cyl r/w
PREX: LD A,D      ;Get MSB of count
      AND 01      ;Ignore all but least sig .5K
      LD B,A      ;BC=corrected count
      LD C,E
      PUSH HL     ;Save HL
      LD HL,0200  ;HL=.5K
      AND A       ;Reset carry
      SBC HL,BC   ;HL=.5K-DE
      EX (SP),HL  ;Restore HL and put calc on stack
      EXX         ;Get alt regs
      POP DE      ;DE'=# of bytes to pad a .5K cyl
      LD BC,01BF  ;Sig need to pad & port BF for data
      LD A,D      ;Get MSB of pad amount
      CP 02       ;Exactly .5K?
      JR NZ,L.5K  ;Jump if not
      XOR A       ;Correct for 0 bytes if already a cyl
      LD D,A
L.5K  OR E        ;Test LSB of pad amount
      JR NZ,MR_0  ;Leave flag alone if pad is needed
      INC B       ;Make flag show no pad req
MR_0  EXX         ;Put back set up alt regs
      RET

EER?  LD A,(SIZE) ;Are we within an ERASE?
      AND A
ZRET  JR Z,SIZE0  ;Skip if not
      IN A,(9F)   ;Get current controller track#
      AND A       ;Skip if CAT saved last
      JR Z,SIZE0
      CALL LCAT   ;Load the CATalog
      CALL U&SC   ;Update and save the CAT

;Here to sound bells and reset sysv SIZE
SIZE0 CALL TOOT  ;Make 7 bell tinks
SIZ0' LD A,00    ;A=00
      LD (SIZE),A ;sv SIZE=0
      RET

;LET /S=n command handler
LET/S CALL AFTEREQ ;Insure next char is "=" & advance past
      CALL EVA#   ;Eval as number, ret in syntax time
      LD A,B      ;Get MSB
      AND A       ;Zero?
```

```
      JP NZ,ER_B  ;Error <>0t
      LD A,C      ;Get LSB
      AND A       ;Zero?
      JP Z,ER_B   ;Error if 0
      CP 03       ;1 or 2?
      JP NC,ER_B  ;Error if neither
      LD (SIDS),A ;Set sv SIDS as directed
      XOR A       ;A=0
      LD (SID#),A ;Side 0
      CALL FND#   ;Form new DSEL
      OUT (B7),A  ;Send it
      JP RETB     ;Ret to rom

;Here to parse into Basic statement following a LET /
LET/  RST 10H     ;Adv to next Basic char
      RES 5,A     ;Lower case=upper case
      CP 'T'      ;T?
      JR Z,LET/T
      CP 'S'      ;S?
      JR Z,LET/S
      CP 'D'      ;D?
      JR Z,LET/D
      CP 'P'      ;P?
      JP Z,LT/P
      CP 'H'      ;H?
      JP NZ,REER  ;Real error if none of above

;LET /H command handler
LET/H CALL AFTEREQ ;Insure next char is "=" & advance past
      CALL EVA#   ;Eval as a #, error if not. Ret to rom in syn
      LD A,B      ;Get MSB
      AND A       ;Zero?
      JP NZ,ER_B  ;Error B if not
      LD A,C      ;Get LSB
      CP 04       ;<4?
      JP NC,ER_B  ;Error B if not
      LD (HDSP),A ;Set SV HDSP (head speed) as directed
      JP RETB     ;Ret to rom

;LET /D command handler
LET/D CALL AFTEREQ ;Insure "=" is next char & advance past
      CALL EVA#   ;Eval as #, ret to rom in syn time
      LD A,B      ;Get MSB
      AND A       ;Zero?
      JP NZ,ER_W  ;Error W if not (Invalid drive#)
      LD A,C      ;Get LSB
      CP 04       ;<3?
      JP NC,ER_W  ;Error W if not from 0-3
      LD (DNUM),A ;Set selected drive#
```

```
      CALL FND#   ;Find new DSEL for this drive
      OUT (B7),A  ;Send it
      JP RETB     ;Ret to rom

;Here to handle LET/T
LET/T CALL AFTEREQ ;Insure next char is "=" & advance past
      CALL EVA#   ;Error if not number & ret in syntax time
      LD A,B      ;MSB of number to A
      AND A       ;Number > 255?
      JR NZ,ER_B  ;Error B if it is
      LD A,C      ;LSB of number to A
      CP 02       ;Less than 2 tracks?
      JR C,ER_B   ;Out of range if so
      LD (TRKS),A ;Store new tracks setting

;Here to clear 3 words from stack & ret to Basic rom
RETB: POP HL      ;Clear stack
      POP HL
      POP HL
;Now step-in disk head if drive motor on & at track#0
RETB2 PUSH AF     ;Save AF
      IN A,(9F)   ;Get current track#
      AND A       ;Track 0?
      JR NZ,LEAVE ;Skip step-in if not track 0
      IN A,(8F)   ;Get controller status
      RLA         ;Motor on bit to carry
      JR NC,LEAVE ;Skip step-in if motor not running
      CALL REDY   ;Wait till controller ready
      LD A,50     ;50H is head step-in command
      CALL SEND8F ;Send step-in command
      CALL REDY   ;Insure step-in is finished before leaving
LEAVE POP AF      ;Restore AF
      RST 18H     ;Ret to Basic rom

ER_3  LD L,02     ;Error 2
      JR ER_L     ;Jump

ER_B  LD L,0A     ;Error "B"

;Here for error#=Reg L + 1
ER_L  LD DE,(chad) ;Get current char addr
      LD (xptr),DE ;Now also error addr
      JP ENTL     ;Ret w/error

;Here to check that ENTER or ":" ends each statement
;Error C if not
CKND  LD HL,0018  ;Prepare HL
      PUSH HL     ;Address 18H on stack
      RST 20H     ;Call 0018H in Basic rom (get cur char)
```

```
CKN2  CP 0D       ;Character ENTER?
      RET Z       ;OK if it is
      CP ':'      ;Character ":"?
      RET Z       ;OK if it is, too

;Error C
ER_C  LD L,0B     ;Show error C
      JR ER_L

;Okidata printer COPY /
OKCP  LD A,03     ;Send a 3 to printer (enter graphics mode)
      CALL PRTA
      LD BC,BF00  ;Start at pixel 0,191
      LD H,1C     ;28d graphics lines to send
OKL3  LD L,B      ;Store Y coordinate in L
OKL2  LD B,L      ;Get Y coordinate
      LD D,07     ;Each graphics line is 7 dots high
OKL1  PUSH HL     ;Save used registers
      PUSH AF
      PUSH BC
      LD A,BF     ;Set A for test
      SUB B       ;Pointing lower than screen?
      CCF         ;Reset carry (pixel not set) if off screen
      JR NC,RETP  ;Skip with pixel not set if off screen
      CALL PXAD   ;Get pixel address into HL
      LD B,A      ; Get byte pointed
      INC B       ; to by HL, find
      LD A,(HL)   ; correct bit for
OKL4  RLCA        ; pixel & put
      DJNZ OKL4   ; it in the
      RRA         ; carry flag
RETP  RL H        ;Save pixel
      POP BC      ;Restore BC
      POP AF      ;Restore AF
      RR H        ;Restore pixel
      POP HL      ;Restore HL
      RRA         ;Add pixel to A
      DEC B       ;One pixel lower
      DEC D       ;Dec dot counter
      JR NZ,OKL1  ;Loop till 7 pixels fetched
      AND A       ;Clear carry
      RRA         ;Pixels to lower 7 bits
      CALL PRTA   ;Send pixels to printer
      CP 03       ;Was pixel code the OKIDATA graphics code?
      CALL Z,PRTA ;Send it again if it was
      INC C       ;Point at next column
      JR NZ,OKL2  ;Loop till 256 bytes sent
      LD A,03     ; Send a
      CALL PRTA   ; graphics
```

```
      LD A,0E     ; line feed/
      CALL PRTA   ; car return
      DEC H       ;Dec graphics line count
      JR NZ,OKL3  ;Loop till all graphics lines sent
      LD A,03     ;Send a code 3 then 2 to exit graphics mode
      CALL PRTA
      LD A,02
      JR PRTA     ;Will ret via PRTA

;Here to send char in A to printer
ORG 033C          ;Must keep PRTA @033C 'cause documented
PRTA  PUSH AF     ;Save char to send
BRK?  CALL TBRK   ;Error C if Break pressed
      IN A,(7F)   ;Get printer status
      BIT 4,A     ;Printer busy?
      JR NZ,BRK?  ;Loop if it is
      POP AF      ;Restore char to send
      OUT (7F),A  ;Send it
      RET

;This is data sent to the Gemini printer in its driver
GMDATA DB 1B,4D,14,1B,31
DAT2   DB 1B,4B,00,01

;Here to restore to track 0, side 0, and form DSEL
RS02  CALL REDY   ;Wait till controller ready
RSR0  XOR A       ;A=00
      LD (SID#),A ;Side 0 is now current
      CALL FND#   ;Form new DSEL
      JR RSTR     ;Restore drive

;Here to restore current drive to track 0
RSR2  CALL REDY   ;Wait till controller ready
;Enter here to restore w/o busy check
RSTR  LD A,(DSEL) ;Get soft copy of port B7
      OUT (B7),A  ;Send it
      LD A,(HDSP) ;Get drive headstep speed
      AND 43      ;Step in a track to insure away from track 0
      CALL SEND8F
      CALL REDY   ;Wait till controller ready
      LD A,(HDSP) ;Get drive headstep speed
      AND 03      ;Strip off other bits to select restore com
      CALL SEND8F ;Send command to controller
MTR?  IN A,(8F)   ;Get controller status
      BIT 5,A     ;Motor made 6 revolutions?
      RET NZ      ;Done if it has
      CALL TBRK   ;Ret to Basic rom if Break pressed
      JR MTR?     ;Loop to test status again
```

```
;Gemini screen copy routine
GMCP  LD B,09     ;9 bytes to send
      LD HL,GMDATA ;Point at data to send
SNDT1 LD A,(HL)   ;Get byte
      INC HL      ;Point to next
      CALL PRTA   ;Send byte
      DJNZ SNDT1  ;Cont till 9 bytes sent
      LD BC,BF00  ;Start at pixel 0,191
      LD H,1C     ;7 bit graph line count=28d
GML3  LD L,B      ;Store Y coordinate in L
GML2  LD B,L      ;Get Y coordinate
      LD D,07     ;Set dot counter to 7 dots
GML1  PUSH HL     ;Save main registers
      PUSH AF
      PUSH BC
      LD A,BF     ;Prepare for test
      SUB B       ;Test Y coord>191 (wrapped at bottom)
      CCF         ;Reset carry if it is (dot not plotted)
      JR NC,REPR  ;Skip if dot off of screen
      CALL PXAD   ;Else get pixel address into HL
      LD B,A      ; Get byte pointed
      INC B       ; to by HL, find
      LD A,(HL)   ; correct bit for
GML4  RLCA        ; pixel and put
      DJNZ GML4   ; it in the
      RRA         ; carry flag
REPR  RL H        ;Save pixel in H
      POP BC      ;Restore BC
      POP AF      ;Restore AF
      RR H        ;Pixel back to carry
      POP HL      ;Restore HL
      RLA         ;Add pixel to A
      DEC B       ;One pixel lower
      DEC D       ;Another pixel fetched
      JR NZ,GML1  ;Loop till 7 pixels retrieved
      AND 7F      ;Reset unused bit 7
      CALL PRTA   ;Send the pixel data to printer
      INC C       ;Point at next column
      JR NZ,GML2  ;Loop till 256 bytes sent
      LD A,0A     ;10d is code for a line feed
      CALL PRTA   ;Send a line feed command
      DEC H       ;Dec graphics line count
      JR NZ,GO_ON ;Loop till all graphics lines sent
      LD A,1B     ;Send a 1BH code
      CALL PRTA
      LD A,40     ;Send a 40H code
      JP PRTA     ;And return via PRTA
GO_ON PUSH BC     ;Save BC
      PUSH HL     ;Save HL
```

```
      LD B,04     ;4 data bytes to send
      LD HL,DAT2  ;Point at data
SNDT2 LD A,(HL)   ;Get the data byte
      INC HL      ;Bump
      CALL PRTA   ;Send the byte
      DJNZ SNDT2  ;Loop till 4 bytes sent
      POP HL      ;Restore HL
      POP BC      ;Restore DE
      JR GML3     ;Loop to send next graphics line

;Ret to Basic rom in syntax time only
SYRT  CALL SYN?   ;Are we in syntax time?
      RET C       ;Ret to caller if not
      POP AF      ;Trash caller's ret address

;Here to clear stack & ret to Basic rom
CLST  POP HL      ;Remove last 3 addr from stack
      POP HL
      POP HL
SYNF  LD A,(T/SP) ;Get Timex/Sinc flag
      LD DE,1BEE  ;Point at cknd in Spec rom
      AND A       ;Test flag
      JR NZ,SPEC  ;Jump if Spec
      LD DE,1B44  ;Point at cknd in Timex rom
SPEC  PUSH DE     ;Put cknd addr on stack
      JP RETB2    ;Perform step-in if req then goto cknd in rom

;Write a single track in format time
WRTR  EXX         ;Access alternate registers
      LD C,BF     ;Port BFH
      LD A,H      ;A=H'
      EXX         ;Ret to main register set
      LD HL,SKTP  ;Point within sector # table
      AND A       ;Clear carry
      RLA
TEST  CP 0A
      JR C,CON2
      SUB 0A
      JR TEST
CON2  LD B,A
      LD A,L
      SUB B
      LD L,A
      CALL REDY   ;Wait till controller ready
      LD A,F0     ;Write track command
      OUT (8F),A  ;Send command to controller
      LD BC,1DBF  ;1D=loop count, Port BF
      LD DE,4E0A  ;Byte to send=4E
TWL1  RST 30H     ;Send one         **Total 60d 4Es
```

```
        RST 30H     ;Send one
        DJNZ TWL1   ;Loop for total 1D*2 4Es
        RST 30H     ;Send another
        RST 30H     ;And another
        LD D,B      ;Byte=00
        RST 30H     ;Send 00
        LD B,05     ;Loop count=05    **Total 12d 00s
TWL2    RST 30H     ;Send 00
        RST 30H     ;Send 00
        DJNZ TWL2   ;Total=05*2 00s
        RST 30H     ;Send another 00
        LD D,F5     ;Byte=F5H         **Total 3d F5s
        RST 30H     ;Send F5
        RST 30H     ;Send F5
        RST 30H     ;Send F5
        LD D,FE     ;Byte=FEH         **ID address mark
        RST 30H     ;Send FE
        EXX
        LD D,H      ;Send track#
        RST 30H
        LD D,L
        RST 30H
        EXX
        LD D,(HL)
        RST 30H
        LD D,02     ;Byte=02H
        RST 30H     ;Send 02
        LD D,F7     ;Byte=F7H
        RST 30H     ;Send F7
        LD D,4E     ;Byte=4EH
        RST 30H     ;Send 4E
        LD B,0A     ;0AH loops
TWL3    RST 30H     ;Send 4E
        RST 30H     ;Send 4E
        DJNZ TWL3   ;Total=0AH*2 4Es
        RST 30H     ;Send another 4E
        LD D,B      ;Byte=00
        RST 30H     ;Send 00
        LD B,05     ;05 loops
TWL4    RST 30H     ;Send 00
        RST 30H     ;Send 00
        DJNZ TWL4   ;Total=05*2 00s
        RST 30H     ;Send 00
        LD D,F5     ;Byte=F5H
        RST 30H     ;Send F5
        RST 30H     ;Send F5
        INC HL      ;Point next sector#
        RST 30H     ;Send F5
        LD D,FB     ;Byte=FBH
```

```
      RST 30H     ;Send FB
      LD D,E5     ;Byte=E5H
      RST 30H     ;Send E5
      DEC B       ;FFH loops
TWL5  RST 30H     ;Send E5
      RST 30H     ;Send E5
      DJNZ TWL5   ;Total=FF*2 E5s
      RST 30H     ;Send E5
      LD D,F7     ;Byte=F7H
      RST 30H     ;Send F7
      LD D,4E     ;Byte=4EH
      RST 30H     ;Send 4E
      RST 30H     ;Send 4E
      LD B,08     ;08 loops
TWL6  RST 30H     ;Send 4E
      RST 30H     ;Send 4E
      DJNZ TWL6   ;Total=08*2 4Es
      RST 30H     ;Send 4E
      LD B,E
      RST 30H
      RST 30H
      DJNZ NOTDONE
      RST 30H
      RST 30H
      RST 30H
RDY?  IN A,(8F)
      RRA
      RET NC
      RRA
      JR NC,RDY?
      OUT (C),D
      JR RDY?
NOTDONE RST 30H
      RST 30H
      LD E,B
      RST 30H
      LD D,00
      RST 30H
      RST 30H
      LD B,03
      RST 30H
      RST 30H
      RST 30H
      JP TWL2

;Sector tables, used during FORMAT
STBL DB 01,02,03,04,05,06,07,08,09,0A
SKTP DB 01,02,03,04,05,06,07,08,09,0A
```

```
;Here to set DSEL correctly using (DNUM) & (SID#)
FND#  LD A,(DNUM) ;Get current drive#
      INC A       ;Form bit slot for correct drive
      LD B,A      ;Drive# to B
      XOR A       ;All bits cleared to start
      SCF
ROTA  RLA         ;Rotate set bit into cleared byte
      DJNZ ROTA   ;Set bit (DNUM)
      LD B,A      ;Store result in B
      LD A,(SID#) ;Get current side#
      AND A       ;Side 0?
      JR Z,SID0   ;Jump if side 0
      LD A,80     ;Set bit 7 for side 1
SID0  OR B        ;OR side select bit 7 to drive select bit
      LD (DSEL),A ;Store in DSEL
      RET

;Here to save src drive settings (MOVE)
SVSC  LD A,(SID#) ;Get current side#
      LD (SRSD),A ;Save it
      LD A,(DSEL) ;Get current DSEL
      LD (DSL'),A ;Save it
      CALL REDY   ;Wait till drive ready
      IN A,(9F)   ;Get current track# from controller
      LD (STK#),A ;Save it
      RET

;Here to save dest drive settings (MOVE)
SVDS  LD A,(SID#) ;Get current side#
      LD (DESD),A ;Save it
      LD A,(DSEL) ;Get current DSEL
      LD (DSL2),A ;Save it
      CALL REDY   ;Wait till drive ready
      IN A,(9F)   ;Get current track# from controller
      LD (DTK#),A ;Save it
      RET

;Here to select src drive (MOVE)
SELS  LD A,(SDV#) ;Get src drive#
      LD (DNUM),A ;Make it current drive#
      LD A,(SSDS) ;Get src drive max sides
      LD (DSDS),A ;Make current drive's max sides match
      CALL REDY   ;Wait till controller ready
      CALL SLSR   ;Form new DSEL and set it
      OUT (9F),A  ;Send current track# to controller
      RET

SLSR  LD A,(SRSD) ;Get src side#
      LD (SID#),A ;Now current side#
```

```
      LD A,(DSL') ;Get alt DSEL
      LD (DSEL),A ;Now current DSEL
      OUT (B7),A  ;Send new DSEL
      LD A,(STK#) ;Get src track#
      LD (TRK#),A ;Now current track#
      RET

;Here to select dest drive (MOVE)
SELD  LD A,(DDV#) ;Get dest drive#
      LD (DNUM),A ;Make it current drive
      CALL REDY   ;Wait till controller ready
      CALL SLDE   ;Form new DSEL and set it
      OUT (9F),A  ;Send current track# to controller
      RET

SLDE  LD A,(DESD) ;Get current dest side#
      LD (SID#),A ;Make current side# dest side#
      LD A,(DSL2) ;Get dest drive DSEL
      LD (DSEL),A ;Make current DSEL match
      OUT (B7),A  ;Also send new soft copy to port
      LD A,(DTK#) ;Get current dest track#
      LD (TRK#),A ;Make current track# match
      RET

;FOR / command handler
FOR/  RST 10H     ;Advance to next Basic char
      CALL RSTK   ;Reset stack
      CALL NXEX   ;Evaluate next expression
      JP NC,ER_C  ;Err C if not numeric
      CALL SYN?   ;Syntax time?
      JR NC,SYPT  ;Jump in syntax time
      POP HL      ;Correct stack pointer
      LD HL,(vars) ;Get pointer to variables
      LD A,(HL)   ;Get first byte in variables area
      AND E0      ;Mask don't care bits
      CP 60       ;Simple single char variable?
      JP NZ,ER_X  ;Error X if not
      LD (LPCN),BC ;Store FOR/NEXT loop limit
      INC HL      ;Point past variable name
      EX DE,HL    ;Save pointer in DE
SYPT  LD HL,0018  ;Get current Basic character
      PUSH HL
      RST 20H
      CP CC       ;Is it token for TO?
      JR Z,NDEF   ;Jump if it is, no defaults
      LD BC,0001  ;Make default start 1
      CALL SYRT   ;Ret to rom if in synt time & using defaults
NDEF  CALL SYN?   ;Syntax time?
      JR NC,SYN2  ;Jump in syntax time
```

```
       LD A,00     ;Let A=0
       EX DE,HL    ;Variable pointer back to HL
       LD (HL),A   ;Insure an integer variable
       INC HL
       LD (HL),A
       INC HL
       LD (HL),C   ;Set variable's start value
       INC HL
       LD (HL),B
       INC HL      ;Last byte of variable is 0, too
       LD (HL),A
SYN2   POP DE      ;Back up stack one more word
       JR NZ,DEFL  ;Jump if using assumed default of 1
       RST 10H     ;Advance to next Basic char
       CALL EVA#   ;Eval as a number, ret to Basic in synt time
       LD (LPCN),BC ;Set initial non-default loop limit
DEFL   CALL CKND   ;Error if ends w/other than ENTER or ":"
       LD HL,(chad) ;Get address of current statement
       LD (NXC'),HL ;Save address
       LD A,FF     ;Let A=FFH
       LD (FORF),A ;Signal a valid FOR/ in progress
       LD HL,(ppc_) ;Get Basic's prog counter
       LD (CPP'),HL ;Save it
       LD HL,(nxln) ;Get address of next basic line
       LD (NPA'),HL ;Save it
       LD A,(sbpc) ;Get statement# within line
       LD (CPS'),A ;Save it
       POP HL      ;Ret to Basic rom
       POP HL
       RST 18H

;Main FORMAT routine
MFRM   CALL RCNT   ;Reset retry counter
       CALL RSR0   ;Restore to track 0, side 0, w/o busy check
       EXX
       PUSH HL
       LD HL,0000
       EXX
SND1   CALL WRTR   ;Write a single cylinder
       JR NC,CHEK  ;Jump if no errors
ELOP   CALL RTRY   ;Tink & stop if too many retries
       JR SND1     ;Loop to write again
CHEK   CALL VFCY   ;Verify cyl just created
       JR NC,ELOP  ;Loop for retry if error
       CALL RCNT   ;Reset error counter
       LD A,(SIDS) ;Get # of sides on disk
       DEC A       ;Only one side?
       JR Z,NXTR   ;Jump to adv to next track if single side
       LD A,(SID#) ;Get current side#
```

```
      XOR 01      ;Flop sides
      LD (SID#),A
      EXX         ;Access alternate regs
      LD L,A      ;Save side# for use later
      EXX         ;Back to normal regs
      PUSH AF     ;Save side# flag
      CALL FND#   ;Form DSEL from DRV# and SID#
      OUT (B7),A  ;Send DSEL to controller
      POP AF      ;Restore side# flag
      JR NZ,SND1  ;Loop to send side 1 if needed
NXTR  EXX
      LD A,(TRKS)
      INC H
      CP H
      EXX
      JR NZ,OK??
      EXX
      POP HL
      EXX
      RET
OK??  CALL _1SID  ;Step in and pause 28ms
      JR SND1     ;Loop to write next track

;Store system variables
STSV  OUT (FF),A  ;Clear video port
      IM 1        ;Mode 1 interrupts
      LD SP,7FFF  ;Put stack at top of 16K ram
      CALL REST   ;Perform RESTORE /S
DNEW  LD HL,0000  ;Set to jump to location 0 in rom
      PUSH HL     ;Address 0 to stack
      RST 18H     ;Jump to 0 in rom

;Default main disk system variables
SYSVAR DB 00,DFTRACKS,DFSIDES,01,00,00,DFSTEP,00
       DB 00,01,00,00,COPYFLAG,00,00,00

;Restore default disk system variables
REST  LD DE,DNUM  ;Move default disk svs to B ram
      LD HL,SYSVAR
      LD BC,0010
      LDIR
      LD C,70     ;Allow time for Disk B reset to clear
IDLE  DJNZ IDLE   ;(non-Rev. A Disk B board)
      DEC C
      JR NZ,IDLE
      LD A,F7     ;Set to read keys 1-5
      IN A,(FE)   ;Read them
      OR E0       ;Mask unused bits
      CP FF       ;Any pressed?
```

```
      JR Z,DEFAULT ;Jump to use default if non pressed
      SCF          ;Form new DSEL from key pressed
      RLA
      CPL
      LD (DSEL),A ;Store newly formed DSEL
      DEC B       ;Now find out what this drive# is
NEXT_D INC B
      RRA
      JR NC,NEXT_D
      LD A,B
      LD (DNUM),A ;Store the new non-default drive#
DEFAULT LD A,(DSEL) ;Now send DSEL to the hardware
      OUT (B7),A
      XOR A       ;Clear FOR / flag
      LD (FORF),A
;Now decide if Spectrum or Timex rom is in control
      LD (T/SP),A ;Assume Timex rom
      LD HL,0095  ;Get byte at 0095H in rom into reg L
      CALL GTHL
      LD A,BF     ;Is it BFH?
      CP L        ;A Spectrum rom will have BFH here
      JR Z,SET_SP ;Set Spectrum rom mode if Spec rom found
      LD A,01     ;See if Spectrum emulator present
      OUT (F4),A  ;Turn on Dock bank chunk zero
      LD L,95     ;Get byte at address 0095 in Dock
      CALL GTHL
      XOR A       ;Turn Dock bank back off
      OUT (F4),A
      LD A,BF     ;Is it BFH?
      CP L        ;A Spectrum emulator will have BFH here, too
      RET NZ      ;Leave set as Timex rom if no emulator
SET_SP LD A,FF    ;Signify Spectrum rom present
      LD (T/SP),A
      RET

;Sound 1 bell tink
STOT  LD E,01     ;Set for one bell tink
      JR TOT2     ;Jump to sound it
;Sound 7 bell tinks
TOOT  LD E,07     ;Set for seven bell tinks
TOT2  LD BC,0AF6  ;Ten bytes to send out port F5 & F6
      LD HL,STAB  ;Point at sound table
TLOP  DEC C       ;Port F5
      OUTI        ;Send a byte
      INC C       ;Port F6
      OUTI        ;Send a byte
      JR NZ,TLOP  ;Loop till ten bytes sent
      LD B,07     ;We will pause 7*28ms
WAT4  CALL P28M   ;Pause 28ms
```

```
      DJNZ WAT4
      DEC C       ;Turn off the sound channel
      OUTI
      INC C
      OUTI
      DEC E       ;Decrement "tink" counter
      JR NZ,TOT2  ;Loop till E tinks sounded
      RET

;Sound chip data for bell tink
STAB DB 00,4D,08,10,0C,0A,0D,00,07,3E,07,3F

;LET /P handler
LT/P  CALL AFTEREQ ;Insure next char is "=" & advance past
      RES 5,A     ;Make lower case upper case
      CP 54       ;'T' for Timex printer mode?
      JR Z,ALRT   ;Jump if Timex
      CP 4F       ;'O' for Oliger printer mode
      JR NZ,NZERC ;Error C if neither
ALRT  LD C,A      ;Save O or T
      CALL SYN?   ;Syntax time?
      CALL C,SETP ;Set printer mode, but only in run time
      RST 10H     ;Advance to next valid Basic char
      CP 2F       ;'/'?
      JR Z,ART2   ;Jump for further parse if '/'
      CALL CKN2   ;Error if CR or ':' doesn't end statement
      JR END?     ;Jump to end of routine
ART2  RST 10H     ;Get next char after '/'
      RES 5,A     ;Lower case=upper case
      LD C,04     ;4 is code for Gorilla Banana copy
      CP 42       ;'B'?
      JR Z,ZERR   ;Jump if B
      DEC C       ;3 is code for Gemini/Epson copy
      CP 47       ;'G'?
      JR Z,ZERR   ;Jump if G
      DEC C       ;2 is code for Ollivetti copy
      CP 4C       ;'L'?
      JR Z,ZERR   ;Jump if L
      DEC C       ;1 is code for Okidata copy
      CP 4F       ;'O'?
      JR Z,ZERR   ;Jump if O
      DEC C       ;0 is code for Ascii copy
      CP 41       ;'A'?
NZERC JP NZ,ER_C  ;Error C if non of these
ZERR  CALL SYN?   ;Syntax time?
      JR NC,NXCR  ;Skip actually setting mode if syntax time
      LD A,C      ;Copy mode to A
      LD (COPF),A ;Store copy mode flag
NXCR  RST 10H     ;Advance to next valid character
```

```
END?  PUSH HL     ;Put 1 more word on stack
      CALL SYRT   ;Ret to rom in syntax time
      POP HL      ;Clear the word from stack
      JP RETB     ;Return to Basic rom

;See if key "N" pressed on NMI and NEW if so
NEW?  LD A,7F     ;Scan keyrow containing 'N'
      IN A,(FE)   ;Read keyrow
      BIT 3,A     ;Key 'N' pressed?
      JP NZ,TOOT  ;Jump if N key not pressed
      CALL STOT   ;Else make 1 bell tink
      RST 00H     ;And reset

;Set current printer
SETP  LD A,C      ;Get character after "="
      CP 'T'      ;"T" for Timex?
      JR Z,TIME   ;Jump if T to select 2040 printer
      LD HL,(chan) ;Get pointer to start of channel table
      LD DE,000F  ;Set to add for offset
      XOR A       ;No carry & A=00
      ADC HL,DE   ;Add offset
      LD (HL),A   ;Store 00 as LSB of output routine
      INC HL      ;Bump
      LD (HL),5B  ;Output routine is at 5B00
      INC HL      ;Bump
      LD (HL),10  ;Store 10H as LSB of input routine
      INC HL      ;Bump
      LD (HL),5B  ;Input routine is at 5B10
      LD HL,RMHN  ;Point at ram resident code to move
      LD DE,5B00  ;Dest is printer buffer
      LD BC,001D  ;1DH bytes to move to ram
      LDIR        ;Move it
      RET

;Ram based code for SETP
RMHN CALL B_ON   ;Turn on B bank
     CALL POUT   ;Call printer output routine
     LD A,(0008) ;Turn off B bank
     EI          ;Interrupts on
     RET
     DB 00,00,00,00,00
     CALL B_ON   ;Turn on B bank
     CALL P_IN   ;Ready printer busy status
     LD HL,(0008) ;Turn off B bank
     EI          ;Interrupts on
     RET
     DB FF       ;Default print line length is 255
     DB LNFEED   ;Define char to be sent after a CR
```

```
;Here to reselect regular 2040 printer
TIME LD HL,(chan) ;Point at start of chan
     LD DE,000F  ;Add offset to addresses
     ADC HL,DE   ;Form pointer
     EX DE,HL    ;Pointer is dest address
     LD HL,SPAD  ;Assume Spec mode so point to Spec addresses
     LD A,(T/SP) ;Get Timex/Spectrum flag
     AND A       ;Test flag
     JR NZ,SPEK  ;Jump in Specrum mode
     LD HL,TIAD  ;Point to Timex addresses
SPEK LD BC,0004  ;4 bytes to move
     LDIR        ;Move them
;Now clear printer buffer
CLBF LD HL,5B00  ;Point at printer buffer
     LD DE,5B01  ;Dest is start of buffer +1
     XOR A       ;A=00
     LD (HL),A   ;Clear first byte
     DEC C       ;255 more bytes to clear
     LDIR        ;Clear the rest of the buffer
     RET

SPAD DB F4,09,C4,15 ;Spectrum mode printer I/O addresses

TIAD DB 00,05,BF,11 ;Timex mode printer I/O addresses

;Code here (P_IN & POUT) is acessed from code in printer buffer
P_IN IN A,(7F)   ;Read printer status
     BIT 4,A     ;Test busy bit
     SCF         ;Signal INKEY$ response
     LD A,42     ;Assume 'B'usy
     RET NZ      ;Done if busy
     LD A,52     ;Signal 'R'eady
     RET

;Here to send output to printer
POUT CP 20       ;Code 20H or greater?
     JR NC,CNT2  ;Jump if it is
     CP 06       ;Code less than 6?
     JR C,PRT?   ;Print '?' if it is
     JR Z,HNDC   ;Goto comma handler if code is 6
     CP 18       ;Code from 18-1Fh?
     JR NC,PRT?  ;Print "?" if it is
     CP 0C       ;Delete code?
     JR NZ,CNT1  ;Skip if not
     BIT 4,(IY+01) ;(IY+1=flgs) (Are we in command mode?)
     RET NZ      ;Print nothing and rt if not
     LD A,(T/SP) ;Get Timex/Spectrum flag
     AND A       ;Test flag
     RET NZ      ;Done if in Spectrum mode
```

```
     LD A,7A     ;Set A for DELETE code
     JR PTKZ     ;Print the token
CNT1 CP 16       ;Code for AT control?
     JR Z,HNAT   ;Handle AT if it is
     CP 17       ;Code for TAB control?
     JR Z,HTAB   ;Handle TAB if it is
     CP 0D       ;CR?
     JR Z,CNT3   ;Jump if a carriage return
     CP 10       ;Code <16d?
     RET C       ;Print nothing & return if it is
SLP1 LD DE,RECC  ;Here for color codes, etc. (Skip 1 byte)
CHAD LD HL,5B04  ;Point at current channel address pointer
     LD (HL),E   ;Change channel address as required
     INC HL      ;And return
     LD (HL),D
     RET
RECC LD DE,POUT  ;Point at normal P_OUT address
     JR CHAD     ;Change channel address back
;Here to print a '?' character
PRT? LD A,3F     ;Set code for '?' character
     JR CNT3     ;Print it
HNAT LD DE,ATCN  ;Point current channel add to AT control
     JR CHAD     ;Return via change channel address routine
ATCN LD DE,XXX3  ;Point current channel at XXX3
     JR CHAD     ;And return
XXX3 CALL RECC   ;Restore current output address to norm
XXX4 LD E,A      ;E=new col position
XXX5 LD A,(ppos) ;A=current col position
     CP E        ;New position < old position?
     RET Z       ;Return if a match
     JR NC,XXX6  ;Send a CR if new pos < old position
     LD A,20     ;20h is ASCII space code
LP_1 CALL PRNT   ;Print the character
     JR XXX5     ;Loop to test again
XXX6 LD A,0D     ;0D is code for CR
     JR LP_1     ;Jump to print it & retest
HNDC LD A,(ppos) ;Get current print column position
     AND F0      ;Reset bits 0-3
     ADD A,10    ;Form position of next TAB field
     JR XXX4     ;Go TAB to next field
HTAB LD DE,TAB2  ; Point current channel to TAB2
     JR CHAD     ; and return via chcn
TAB2 CALL XXX4   ;TAB to parimeter supplied
     JR SLP1     ;Return, but skip next parimeter
CNT2 LD C,A      ;Save code
     LD A,(T/SP) ;Get Timex/Spectrum flag
     AND A       ;Spectrum mode?
     LD A,C      ;Restore code
     JR NZ,CNT3  ;Skip in Spectrum mode
```

```
     CP 7C       ;STICK token?
PTKZ JR Z,PTOK   ;Handle it if it is
     CP 7E       ;FREE token?
     JR Z,PTOK   ;Handle it if it is
     CP 7B       ;Is code a token at all?
     JR C,CNT3   ;Skip if its not
     CP 80       ;Is code a non-Spectrum token?
     JR NC,CNT3  ;Skip if not Timex token
     BIT 4,(IY+01) ;(IY+01=flgs) Are we in command mode?
     JR Z,PTOK   ;Print token if we are
CNT3 CP 80       ;Is code not a token/graph character?
     JR C,PRNT   ;Send character if not
     CP 90       ;Is it a block graph char?
     JR NC,TOUG  ;Print token or UDG if not
     LD A,20     ;Set to print a space
     JR PRNT     ;Print a space and return
TOUG SUB A5      ;Forn offset for token look-up
     JR NC,PTOK  ;Print token if not UDG
     ADD A,56    ;Make UDG codes letters A-U
     JR PRNT     ;Print letter
PTOK LD DE,0095  ;Point at Spectrum token table
     LD HL,0C41  ;Point at Spectrum token look-up routine
     PUSH AF     ;Save code
     LD A,(T/SP) ;Get Timex/Spectrum flag
     AND A       ;Test flag
     JR NZ,SINN  ;Skip if Spectrum mode in effect
     LD DE,0098  ;Point at Timex token table
     LD HL,077C  ;Point at Timex token look-up routine
SINN POP AF      ;Restore code
     CP 5B       ;Is it a Spectrum token?
     JR C,SIN2   ;Skip if it is
     SUB 1F      ;Correct offset for Timex token
SIN2 PUSH AF     ;Save token look-up code
     PUSH HL     ;Token look-up routine address to stack
     RST 20H     ;Call look-up routine
     JR C,TOCH   ;Go print token if no leading space
     BIT 0,(IY+01) ;(IY+01=flgs) Do we need a leading space?
     JR NZ,TOCH  ;Print token if we don't
     LD A,20     ;20H is ASCII space
     CALL PRNT   ;Print a space
TOCH EX DE,HL   ;Exchange pointers
TCH1 PUSH HL    ;Save pointer
     CALL GTHL  ;Get a character of token text
     LD A,L     ;Put character in A
     EX DE,HL   ;Exchange pointers back
     AND 7F     ;Insure bit 7 is reset
     CALL PRNT  ;Print the character
     LD A,E     ;Get the character again
     POP HL     ;Restore pointer
```

```
     INC HL     ;Point at next character
     ADD A,A    ;Test bit 7
     JR NC,TCH1 ;Keep printing characters till bit 7 set
     POP DE     ;Restore look-up code
     CP 48      ;Was last char "$"?
     JR Z,TRSP  ;Print trailing space if it was
     CP 82      ;Last char any before "A"?
     RET C      ;Ret if it was, no trailing space
TRSP LD A,D     ;Token look-up code to A
     CP 03      ;RND, INKEY$, or PI?
     RET C      ;No trailing space if it was
     LD A,20    ;Load code for space & print it
PRNT LD HL,flgs ;Point at Basics flag sys variable
     RES 0,(HL) ;Signal no space this time
     CP 20      ;But is it a space?
     JR NZ,SNDA ;Send it if not
     SET 0,(HL) ;Signal a space this time
SNDA PUSH AF    ;Save code to send
     LD HL,ppos ;Get printing position
     CP 0E      ;Position less than 14d?
     JR C,O.K.  ;Skip is it is
     LD A,(5B1B) ;Get max print col position
     CP (HL)    ;Reached it yet?
     JR NZ,O.K. ;Skip if not
     LD A,0D    ;0Dh is ASCII CR
     CALL SNDA  ;Send a CR to printer
O.K. INC (HL)   ;Bump to next print position
STP? CALL TBRK  ;Test for BREAK key pressed & stop if it is
     IN A,(7F)  ;Get printer status
     BIT 4,A    ;Printer busy?
     JR NZ,STP? ;Loop if it is
     POP AF     ;Restore code to send
     OUT (7F),A ;Send the code to the printer
     CP 0D      ;Did we just send a CR?
     RET NZ     ;Done if not
     XOR A      ;A=00
     LD (HL),A  ;Reset print col position
     LD A,(5B1C) ;Get code to send after a CR
     AND A      ;Null code?
     RET Z      ;Send nothing if it is
     JR SNDA    ;Else send the character and return

;SAFE V1 save/load n routine
SL/N CALL CKND  ;Insure CR or ":" end statement
     CALL SYRT  ;Ret to rom in syntax time
     LD A,(SV/L) ;Get SAVE/LOAD/MERGE flag
     AND A      ;Test flag
     JP NZ,LD/N ;Jump in not zero. Loading
     LD A,B     ;Insure save is a SAVE /0
```

```
     OR C
     JP NZ,ER_C ;Error C if SAVE argument <>0

;Save file zero
SAV0 LD HL,(elin) ;Get pointer just past end of vars
     LD DE,(prog) ;Point to start of program
     SCF        ;One less cause first pointer 1 more
     SBC HL,DE  ;Find length of pgm & variables
     JP Z,DONEOK ;Done if file has length of 0
     LD (2000),HL ;Store file size
     LD BC,FA04 ;Test length not too long
     ADD HL,BC
     JR C,ER_V  ;Error V if file size too big
     LD HL,(vars) ;Get pointer to variables
     SBC HL,DE  ;Figure offset to variables
     LD (2002),HL ;Store offset
     CALL RSR0  ;Restore drive
     LD H,D     ;HL=src=(prog)
     LD L,E
     LD BC,(2000) ;Get file size
     LD DE,2004 ;Point to first free buffer location
     LDIR       ;Move the file to Bram
     LD HL,2000 ;Point to start of buffer
     LD DE,0600 ;6 pages in file 0
     CALL SVHD  ;Save file 0
     JP DONE    ;Ret to rom in common code, NC=error

;Error W
ER_W LD A,04    ;New error #4
     JR NEWR    ;Jump to handle new error

;Error V
ER_V LD A,03    ;New error #3
NEWR JP NERR    ;Jump to handle new error

;Reset stack pointer
RSTK POP HL     ;Get ret address
     LD SP,(ersp) ;Set SP to ersp
     LD B,0A    ;SP=(ersp)-5 words
DECR DEC SP
     DJNZ DECR
     LD (SPST),SP ;Save new SP address
     JP (HL)    ;Return

;Here to move filename @DE to TCAT (BC bytes long)
;Truncate if length >10d or pad w/spaces if <10d chars
STST LD A,C     ;Get filename length
     OR B       ;Test null string
     JP Z,ER_F  ;Error F if null string
```

```
     LD HL,FFF5 ;Test length >10d chars
     ADD HL,BC
     JR NC,LNOK ;Skip if not >10d chars
     LD BC,000A ;Truncate to 10 chars
LNOK EX DE,HL   ;Src pointer to HL
     LD DE,TCAT ;Dest is TCAT
     LDIR       ;Move filename
PAD# LD A,E     ;Test 10 bytes long
     CP 2A
     RET Z      ;Done if 10 bytes
     LD A,20    ;Else pad with a space
     LD (DE),A  ;Store space
     INC DE     ;Bump
     JR PAD#    ;Loop till name padded to 10d chars

;Here to turn off exrom & reset bank switch stack pointer
OFEX PUSH AF     ;Save registers
     PUSH DE
     PUSH HL
     LD A,(T/SP) ;Timex or Spec?
     AND A
     JR NZ,EXIT  ;Done if Spec
     LD A,(vmod) ;Get Timex video mode sv
     AND A       ;Mode 0?
     JR Z,VID0
     LD A,8E     ;Get LSB to reset bs SP
     LD (FD8E),A ;Reset pointer
     JR VID1     ;Cont later
VID0 LD A,CE     ;Get LSB to reset bs SP in mode 0
     LD (65CE),A ;Reset it
VID1 IN A,(FF)   ;Get vid/exrom on status
     RES 7,A     ;Insure exrom off
     OUT (FF),A
     IN A,(F4)   ;Get dock/exrom chunk sel status
     RRA         ;Chunk 0 on?
     JR NC,EXIT  ;Done if not
     LD HL,(5CBC) ;Point at syscon table
     LD DE,0008  ;8 bytes into table
     ADD HL,DE
     LD A,(HL)   ;Get this byte
     CP 01       ;LROS present?
     IN A,(F4)   ;Prepare to turn off chunk 0 if no LROS
     RES 0,A
     JR NZ,NLRS  ;Jump if no LROS
     INC HL      ;Point at LROS chunk sel byte
     INC HL
     INC HL
     LD A,(HL)   ;Get it
     CPL         ;Compliment it for port
```

```
NLRS OUT (F4),A  ;Send chunk sel byte
EXIT POP HL      ;Restore registers
     POP DE
     POP AF
     RET

;This subroutine will call the address stored at (SP) in Spec
;mode or (SP+2) in Timex mode. The normal return address will
;Be at (SP+4). Reg DE is destroyed here and other regs may be
;destroyed by called routine (normally in another bank).

CALLS/T: EX (SP),HL ;Point at byte after call
     PUSH AF    ;Save AF
     LD A,(T/SP) ;Get Timex/Spectrum flag
     AND A      ;Test flag
     JR NZ,SPECAL ;Jump if in Spectrum mode
     INC HL     ;Skip Spectrum address
     INC HL
SPECAL LD E,(HL) ;Get address lo byte
     INC HL     ;Bump
     LD D,(HL)  ;Get hi byte
     INC HL     ;Bump
     JR Z,TIMCAL ;Done if Timex address
     INC HL     ;Skip Timex lo byte
     INC HL     ;Skip Timex hi byte
TIMCAL POP AF   ;Restore AF
     EX (SP),HL ;Restore updated address to SP
     PUSH DE    ;Address to be called onto stack
     RST 20H    ;Call rom at address supplied
     RET        ;Ret to original routine

;Safe fuction dispatcher address table
FUNCTABLE:
#00: DW TBRK   ;Test break key pressed & stop if it is
#01: DW SL@N   ;Save/Load/Merge command (see manual)
#02: DW ERAS   ;Erase command (see manual)
#03: DW NERR   ;New SAFE error report handler
#04: DW ER_L   ;Ret to basic w/old error report L-1
#05: DW CKND   ;Insure CR or ":" end statement, else error C
#06: DW CATL   ;CaTalog routine
#07: DW CAT2   ;CATalog routine w/o catalog load
#08: DW LCAT   ;Load catalog to B bank buffer
#09: DW LCT2   ;Load catalog w/o drive restore
#0A: DW SCAT   ;Save catalog from B bank buffer
#0B: DW SCT2   ;Save catalog w/o drive restore
#0C: DW NXEX   ;Eval basic expression. Carry set if numeric
;& result in BC. Else BC=length & DE=pointer to text string.
#0D: DW SYN?   ;Test syntax time. NC=syntax time
#0E: DW STOT   ;Sound 1 bell tink
```

```
#0F: DW TOOT   ;Sound 7 bell tinks
#10: DW TOT2   ;Sound E bell tinks
#11: DW PRTA   ;Send char A to Oliger printer port
#12: DW FND#   ;Set (DSEL) using (DNUM) & (SID#)
#13: DW STST   ;Move filename @DE to TCAT, BC bytes long
;Truncate or pad filename to 10 characters as required.
#14: DW CALLS/T ;Call address in Timex or spectrum rom. Use
;next dataword for Spectrum rom or 2nd dataword for Timex rom.
;Correct return address so these 2 words are skipped.
#15: DW NXCX   ;Find next track/side using (SSDS) & (SID#)
;Result returned in (SID#) & (TRK#)
#16: DW PJHL   ;Print & right justify (5 digits) # in HL
#17: DW JTEN   ;Print & right justify (2 digits) # in HL
#18: DW MTCH   ;Match filename/type @(DE) to (HL)
#19: DW L5KB   ;Load 5K bytes to B bank buffer (preset drive)
#1A: DW LD5K   ;Load 5K bytes to (HL) (preset drive)
#1B: DW LDHD   ;Load DE bytes to (HL) (preset drive)
#1C: DW LHD2   ;Load DE bytes to (HL) starting w/sector (SEC#)
#1D: DW S5KB   ;Save 5K bytes from B bank buffer (preset drive)
#1E: DW SV5K   ;Save 5K bytes from (HL) (preset drive)
#1F: DW SVHD   ;Save DE bytes from (HL) (preset drive)
#20: DW SHD2   ;Save DE bytes from (HL) starting w/sect (SEC#)
#21: DW REDY   ;Wait till drive ready. Error D if break pressed
#22: DW SEKT   ;Move drive to (TRK#). NZ=error or Z=O.K.
#23: DW RS02   ;Restore current drive to track 0, side 0
#24: DW RSR2   ;Restore current drive to track 0 (SID#=no chng)
#25: DW NMCP   ;Copy screen to Oliger printer port per (COPF)
#26: DW P28M   ;Pause aprox. 28ms
#27: DW CLS!   ;Clear entire screen
#28: DW REST   ;SAFE's RESTORE /S (Restore SAFE to defaults)
#29: DW SETP   ;Set printer to C (C="T" for 2040, else Oliger)
#2A: DW CLOW   ;Clear lower display


;Error T handler continues here
ERT_CNT CALL EER? ;Reset SIZE if needed & sound bell tinks
     LD A,01     ;New error #1
     JP NERR     ;Handle new error

ORG 0A68         ;Entry to LCAT & SCAT are fixed (documented)
;Load CATALOG
LCAT CALL RSR0   ;Restore drive to track 0/side 0
LCT2 CALL RCNT   ;Reset retry counter
LCT3 CALL L5KB   ;Load 5K cylinder to Bbank buffer
     JP C,RCNT   ;Ret via RCNT if no errors
     CALL RTRY   ;1 tink. Error T if no more trys left
     JR LCT3     ;Loop to try again

;Here if disk error occurs to see if another retry is avail
```

```
RTRY CALL STOT   ;Sound 1 bell tink
     LD HL,TRYS  ;Point at counter
     DEC (HL)    ;Dec counter
     RET NZ      ;Ret if not end of count

;Error T, Disk I/O error
ER_T LD A,(SV/L) ;Get SAVE/MERGE/LOAD flag
     CP 80       ;Merge in progress?
     JP Z,RDER   ;Reset if error occurs on merge
     JP ERT_CNT  ;Jump to save space for SCAT ORG

;Save CATalog
ORG 0A8E         ;Save Catalog routine is documented
SCAT CALL RS02   ;Restore drive
SCT2 CALL S5KB   ;Save the CATalog
     JR NC,BRC?  ;Jump to test break key if error
     CALL VFCY   ;Verify good save
     RET C       ;Done if no errors
BRC? CALL TBRK   ;Test break key
     JR SCAT     ;Loop to try saving CAT again

FTRS LD B,A
     LD A,(TRKS)
     SUB 09
     LD D,A
     XOR A
     LD (SID#),A
     JR FRST
NXBL ADD A,0A
FRST JP C,ER_B
     CP D
     JR NC,NXSID
     DJNZ NXBL
     JR SET_
NXSID LD A,(SIDS)
     DEC A
     JP Z,ER_B
     XOR A
     JR FST2
NXB2 ADD A,0A
FST2 CP D
     JP NC,ER_B
     DJNZ NXB2
     LD B,A
     LD A,FF
     LD (SID#),A
     LD A,B
SET_ LD (TRK#),A
     JP FND#     ;Ret via FND# routine
```

```
;Seek track (TRK#)
SEKT CALL REDY   ;Wait till controller ready
     LD A,(TRK#) ;Get track# to seek
     OUT (BF),A  ;Put track# in data register
     LD A,(HDSP) ;Get head step speed
     AND 03      ;Strip bits 2-7
     OR 10       ;Form seek w/o verify code
     CALL SEND8F ;Send seek command to controller
     CALL REDY   ;Wait till controller finished w/seek
     BIT 3,A     ;Test 'not found' bit
     RET         ;Ret. NZ=error & Z=O.K. (& NC)

;Test syntax time. Return result in carry flag
SYN? LD A,(flgs) ;Get Basic's flgs sv
     RLA         ;Syntax flag to carry
     RET         ;Ret w/carry indicating syntax/run

;Multiply # in A by 5 and print result
D&PA LD HL,0000  ;Start w/zero
     LD DE,0005  ;A*5
     AND A       ;A=00?
ADLP JP Z,PTHL   ;Print result if done
     ADD HL,DE   ;Add 5
     DEC A       ;Done?
     JR ADLP     ;Loop to test done

;Verify cylinder
VFCY: LD BC,1400  ;5K bytes for complete cylinder
     LD (OLDE),BC ;Set to check 5K bytes
VFCY1: CALL REDY   ;Wait till controller ready
     LD A,01     ;Start w/sector 1
     OUT (AF),A  ;Write to sector register
     LD A,90     ;This is read sector multiple command
     CALL SEND8F ;Send command to controller
     LD BC,(OLDE) ;Get # of bytes to check
VERFLP: IN A,(8F) ;Get controller status
     RRA         ;Busy bit to carry
     RET NC      ;Signal error if no longer working
     RRA         ;Another byte ready?
     JR NC,VERFLP ;Loop if not
     IN A,(BF)   ;Read & trash the byte
     DEC BC      ;Dec byte counter
     LD A,B      ;Counter zero?
     OR C
     JR NZ,VERFLP ;Loop for next byte if not
;Here to wait for controller to try & access next sector,
;Then a force stop is commanded
WAITSEK: IN A,(AF)   ;Read sector register
```

```
     LD C,A      ;Save current sector#
LOOKLP: IN A,(BF) ;Do a dummy read
     IN A,(8F) ;Get controller status
     RRA         ;Busy?
     JR NC,END   ;Done if controller not busy
LOOK2: IN A,(AF)  ;Read sector reg
     CP C        ;Past last sector?
     JR Z,LOOKLP ;Loop till last sector done
SKDONE: LD A,D0     ;Command to force cont to stop
     CALL SEND8F ;Send force stop command
STAT: IN A,(8F)   ;Get final controller status
     RRA         ;Move bits right
END: AND 2E      ;Ignore bits 0,4,6, & 7
     RET NZ      ;Ret w/o carry if no errors
     SCF         ;Else signal error
     RET

;Handle LOAD /0
HLD0: CALL RSTK   ;Reset the stack pointer
     CALL CKND   ;Insure CR or : ends statement
     CALL SYRT   ;Ret to rom in syntax time
;Load file 0
LD/0 POP AF      ;Unclutter the stack
POP AF
POP AF
POP AF
     CALL LCAT   ;Load the CAT and file 0
     LD A,(2004) ;Get first byte of Basic prgm
     RLA         ;Test bit 7
     JP C,ER_S   ;Error S if no program present
     LD HL,(elin) ;Get pointer 1 byte past end of cur prog
     LD DE,(prog) ;Get pointer to start of prgm
     SCF         ;This will account for extra byte
     SBC HL,DE   ;Form length of current program
     PUSH DE     ;Save (prog)
     EX DE,HL    ;Save length
     LD HL,(2000) ;Get length of file 0 program
     SCF         ;Test length
     SBC HL,DE   ;Is length ok?
     JR C,OKLN   ;Skip if ok
     LD DE,0005  ;Else check enough mem avail +5 bytes
     ADD HL,DE
     LD B,H      ;Bytes needed to BC
     LD C,L
     CALL CALLS/T ;See if enough memory available
     DW 1F05     ;Spectrum memtest routine
     DW 1FBB     ;Timex memtest routine
OKLN POP DE      ;Get pointer to start of program
     LD HL,(elin) ;Get pointer 1 byte past end of prog
```

```
     DEC HL      ;Point to last byte of prog
     LD A,(T/SP) ;Timex or Spec rom present flag
     LD BC,19E5  ;Address of reclaim routine in Spec rom
     AND A       ;Test flag
     JR NZ,SIN3  ;Skip if Spec rom present
     LD BC,174D  ;Else point to Timex reclaim routine
SIN3 PUSH BC     ;Rom address to stack
     RST 20H     ;Call the mem reclaim routine
     LD BC,(2000) ;Get # of bytes needed by file 0
     PUSH HL     ;Save pointer to start of prog
     PUSH BC     ;Save length of file 0
     CALL CALLS/T ;Insert BC bytes starting at HL
     DW 1655     ;Spectrum insert routine
     DW 12BB     ;Timex insert routine
     INC HL      ;Point 1 lower
     LD BC,(2002) ;Get offset to start of vars
     ADD HL,BC   ;Form address
     LD (vars),HL ;Store pointer to variables
     XOR A       ;A=00
     LD L,A      ;HL=0000
     LD H,A
     LD (nwpc),HL ;Clear some Basic svs
     LD (nspc),A ;to force auto-run
     POP BC      ;Restore length of file 0
     POP DE      ;Restore pointer to start of Basic prog
     LD HL,2004  ;Point at start of file 0
     LDIR        ;Move file 0 to basic area
     RST 18H     ;Ret to rom

;Find next track/side
NXCX LD A,(SSDS)
     DEC A
     JR Z,SID1
     LD A,(SID#)
     AND A
     JP Z,NXC2
SID1 LD HL,DSEL
     RES 7,(HL)
     LD L,87
     LD (HL),00
     INC HL
     INC (HL)
     RET

;Load SAFE V1 file "N"
LD/N CP 80       ;Merge?
     JP Z,ER_C   ;Error C if MERGing a V1 file
     LD A,B      ;MSB of file# to A
     AND A       ;File# >255?
```

```
     JP NZ,ER_B  ;Error B if >255
     LD A,C      ;LSB of file# to A
     AND A       ;Zero?
     JP Z,LD/0   ;Jump to load file 0 if it is
     LD SP,37FF  ;Use stack in Bram
     CALL LD50   ;Load 50K
     LD A,(NMIF) ;Get NMI flag
     AND A       ;Test flag
     JP NZ,NMSR  ;Jump if saved via NMI interrupt
     LD SP,(SPST) ;Reset SP
     EXX         ;Access alternate regs
     LD HL,(HL'S) ;Restore HL'
     EXX         ;Back to main registers
     JP RETB     ;Ret to rom

;RESTORE /S handler
;Executes both is Syntax & real time
REST/S RES 5,A   ;Make lower case upper case
     CP 53       ;"S"?
     JP NZ,NXEX  ;Ret to RX/X via NXEX if not S
     POP HL      ;Trash ret address to RX/X
     RST 10H     ;Advance to next Basic char
     CALL CKN2   ;Insure CR or ":" end statement
     CALL REST   ;Initialize SAFE
     JP DONEOK   ;Signal no errors & ret to rom

;Clear lower display
CLOW CALL CALLS/T ;Call Timex or Spectrum rom
     DW 0D6E     ;Spectrum clear lower screen routine
     DW 08A9     ;Timex clear lower screen routine
     RET

;Here to print & right justify number in HL
PJHL LD A,20     ;Print space instead of nothing
     JR PHL2     ;Print number in HL

;Here to print & right justify 2 digit # in HL
JTEN LD A,20     ;Print space instead of nothing
     LD (PFLG),A ;Store flag
     JR PTEN     ;Go & print 2 digit decimal#

;Print decimal # in reg L
PNTL LD H,00     ;# must only be in L (0-255)

;Print decimal number in HL
PTHL XOR A       ;Start by printing nothing for a zero
PHL2 LD (PFLG),A ;Store flag
     LD DE,2710  ;DE=10,000
     CALL PTCO   ;Print 10 thousands
```

```
     LD DE,03E8  ;DE=1,000
     CALL PTCO   ;Print thousands
PHUN LD DE,0064  ;DE=100
     CALL PTCO   ;Print hundreds
PTEN LD DE,000A  ;DE=10
     CALL PTCO   ;Print tens
PONE LD A,30     ;Add ASCII offset for units
     ADD A,L

;Print character in reg A
PNTA LD (STHL),HL ;Save HL
     LD HL,0010  ;Call 0010 in rom (RST10)
     PUSH HL     ;Dest of call to stack
     JP CONT20   ;Print char & return

;Here to print a single space to screen
SPAC LD A,20     ;ASCII code for space
     JR PNTA     ;Print a space on screen

;Load 50K (SAFE V1)
LD50 CALL FTRS   ;Find track# & side# of V1 file
     CALL RSTR   ;Restore drive
     CALL SEKT   ;Seek correct track#
     JP NZ,ER_T  ;Stop w/error T if error
     LD HL,3E00  ;Load to starting address of 3E00
     LD DE,0E00  ;Only 0E00 bytes to load from 1rst cyl
     LD A,04     ;Start w/sector 4
     LD (SEC#),A
     CALL LHD2   ;Load 1rst cylinder
     JR NC,RDER  ;Jump if error
     LD A,(DSDS) ;Save this sv
     PUSH AF
     LD A,01     ;Fool into thinking only 1 side
     LD (DSDS),A
     CALL _1SID  ;Advance to next track
     LD DE,B400  ;Need to load 45K more
     CALL LDLP   ;Load the rest
     EX AF,AF'   ;Save carry flag
     POP AF      ;Restore tampered with sv
     LD (DSDS),A
     EX AF,AF'   ;Restore flags
     RET C       ;Done if no errors

;Read error in loading a STATE file
RDER LD A,D0     ;Forcestop command
     OUT (8F),A  ;Send command
     CALL TOOT   ;Ring some bells
     LD SP,7FFF  ;Stack to top of 16K ram
     XOR A       ;A=00
```

```
     OUT (FF),A  ;Clear video port
     OUT (F4),A  ;Clear dock select port
     JP DNEW     ;NEW w/o clearing SAFE variables

;Evaluate next Basic expression
NXEX: CALL CALLS/T ;Call Spectrum or Timex rom
     DW 24FB     ;Spectrum 'scanning' address
     DW 2854     ;Timex 'scanning' routine
     CALL SYN?   ;Syntax time?
     JP NC,SYNR  ;Skip in syntax time
     RLA
     JP NC,UNSK
     CALL CALLS/T ;Call Spectrum or Timex rom
     DW 1E99     ;Spectrum FIND-INT-2 routine
     DW 1F23     ;Timex FIND-INT-2
     SCF
     RET

;NMI SAVE button press handler
NMSA PUSH AF     ;Save AF
     IN A,(FF)   ;Get video/exrom port
     RLA         ;Exrom active?
     JR NC,NOEX  ;Skip if not
     IN A,(F4)   ;Get dock/exrom chunk sel status
     RRA         ;Exrom enabled?
     JR NC,NOEX  ;Skip if not
     POP AF      ;Restore AF
     LD HL,00E5  ;00E5 to stack
     PUSH HL
     LD L,76     ;0076 to stack
     PUSH HL
     LD HL,1F80  ;Either 1F80 or
     BIT 7,A     ;Test flag
     JR Z,RTEX   ;Jump if 1F80 ok
     LD HL,0C98  ;Else 0C98
RTEX RST 18H     ;Return to EXROM if ORG there
NOEX POP AF      ;Restore AF
NMOK POP HL      ;Restore HL
     LD (SPST),SP ;Save all registers
     LD (HLST),HL
     LD (DEST),DE
     LD (BCST),BC
     POP HL
     LD (AFST),HL
     EX AF,AF'
     PUSH AF
     POP HL
     LD (AF'S),HL
     EXX         ;And all alternate registers
```

```
      LD (HL'S),HL
      LD (DE'S),DE
      LD (BC'S),BC
      LD (IXST),IX
      LD (IYST),IY
      LD A,I      ;Save reg I
      LD (I_ST),A
      LD A,R      ;Save reg R
      LD (R_ST),A
      LD A,FF     ;A=FFH
      LD (NMIF),A ;Show via NMI flag that this is a NMI SAVE
      JP PE,YESINT ;Jump if interrupts enabled
      INC A       ;A=00
YESINT LD (INTF),A ;Flag=00 for no int or FF if int enabled
      IN A,(FF)   ;Save video/exrom control port
      LD (FFST),A
      LD A,1B     ;Mode2 vector=1Bxx
      LD I,A
      XOR A       ;A=00
      OUT (FF),A  ;Insure maskable interrupts on
      EI          ;Enable interrupts
      HALT        ;Wait on interrupt
      LD (MODF),A ;MODF will be FF/mode1 or 00/mode2
      CALL NEW?   ;New if key N pressed
      CALL NMSV   ;Else wait on a key & handle

;Control returns here if NMI file loaded
NMSR LD A,(R_ST) ;Restore reg R
      LD R,A
      LD A,(I_ST) ;Restore reg I
      LD I,A
      LD IY,(IYST) ;Restore all other registers
      LD IX,(IXST)
      LD BC,(BC'S)
      LD DE,(DE'S)
      LD HL,(HL'S)
      EXX
      LD HL,(AF'S)
      PUSH HL
      POP AF
      EX AF,AF'
      LD HL,(AFST)
      LD SP,(SPST)
      EX (SP),HL
      LD BC,(BCST)
      LD DE,(DEST)
      LD HL,(HLST)
      LD A,(FFST) ;Restore video/exrom control port
      OUT (FF),A
```

```
     LD A,(MODF) ;Get interrupt mode flag
     IM 1        ;Assume Mode 1
     AND A       ;Test flag
     JR Z,MODE1  ;Jump if correct
     IM 2        ;Else Mode 2 interrupts
MODE1 LD A,(INTF) ;Get interrupt enable flag
     AND A       ;Test flag
     JP NZ,CALL  ;Ret w/interrupts enabled if required
     LD A,(T/SP) ;Timex or Spectrum rom?
     AND A
     JP NZ,SRNI  ;Ret to Spec rom w/o interrupts enabled
     JP TRNI     ;Or to Timex rom w/o interrupts enabled

;Here to match (DE) to (HL) which checks to see
;if filename/type matches
MTCH LD A,(HL)   ;Get first character of filename
     CP 80       ;End of catalog marker?
     RET Z       ;Ret w/carry reset for no match if EOF
     PUSH DE     ;Save pointer to name to find
     PUSH HL     ;Save pointer to current catalog entry
     LD B,0B     ;11 characters must match (10 chars + type)
NTST LD A,(DE)   ;Get character of name to find
     CP 60       ;Is this character the wildcard?
     JR Z,WILDCD ;Jump if it is. It's considered a match
     CP (HL)     ;Does the cat entry match the name?
     JR NZ,NMTH  ;Loop for next entry if no match
WILDCD INC HL    ;Point to next char of cat entry
     INC DE      ;And next char of name to find
     DJNZ NTST   ;Loop till 11 characters match
     POP HL      ;Restore start of matching cat entry
     POP DE      ;And start of the name it matches
     SCF         ;Show that a match was found & ret
     RET
NMTH POP HL      ;Restore old cat entry address
     POP DE      ;And start of name to find
     LD BC,0014  ;Calc start of next cat entry
     ADD HL,BC
     JR MTCH     ;Loop to check

;Keys 1 through 5
K1_5: LD B,30    ;Start 1 less than code for "1"
NXNM INC B       ;Now is next ASCII #
     RRA         ;Keypress to carry
     JR C,NXNM   ;Loop if that key not pressed
     JR Z1_A     ;Jump...keypress!

;Keys 6 through 0
K6_0: LD B,30    ;Start with ASCII for "0"
     RRA         ;Is it "0"
```

```
     JR NC,Z1_A  ;Jump if it is
     LD B,3A     ;Start now with code for "9" less 1
NXN2 DEC B       ;Next lower ASCII #
     RRA         ;Keypress to carry
     JR C,NXN2   ;Loop if that key not pressed
Z1_A: LD C,04    ;4 is code for STATE file
     LD HL,3E00  ;State file starts @3E00H
     LD (TBEG),HL
     LD H,C2     ;All STATE files are C200H bytes long
K1_A: LD (TLEN),HL
     LD HL,TCAT  ;Point at CAT entry build-up area
     LD (HL),B   ;First char is Ascii letter name
     LD A,20     ;Pad the rest of entry w/spaces
     LD B,09
PADI INC HL
     LD (HL),A
     DJNZ PADI
     INC HL      ;Bump to first byte after name
     LD (HL),C   ;Store file type code here
     XOR A       ;Show that this is a SAVE
     LD (SV/L),A
     CALL SL@N   ;Save the file
     RET C       ;Done if no errors
     CALL TOOT   ;Make 7 bell tinks
     JP TOOT     ;And 7 more...

;SCREEN$ SAVE A through E
ATOE LD HL,4000  ;Screen starts @4000H
     LD (TBEG),HL ;Store start address
     LD B,H      ;Ascii value starts with 'A' -1 (40H)
NXLT INC B       ;Next letter
     RRA         ;Rotate key location to carry
     JR C,NXLT   ;Loop till B=Ascii A-E
     LD H,1B     ;Screen length is 1B00H
     LD C,03     ;3 is code# for a BYTES save (SCREEN$)
     JR K1_A     ;Continue in common code w/other keys

;This routine is called from the NMI SAVE routine
;It scans the keyboard and responds to the correct keypress
NMSV LD HL,FFE0  ;Set HL w/mask & CP#
     LD A,FB     ;Read keys QWERT for SCREEN$ SAVE
     IN A,(FE)
     OR L
     CP H
     JR NZ,ATOE  ;Jump if one pressed
     LD A,F7     ;Read keys 12345 for STATE SAVE
     IN A,(FE)
     OR L
     CP H
```

```
     JR NZ,K1_5  ;Jump if one pressed
     LD A,EF     ;Read keys 67890 for STATE SAVE
     IN A,(FE)
     OR L
     CP H
     JR NZ,K6_0  ;Jump if one pressed
     LD A,BF     ;Read keys HJKL & ENTER
     IN A,(FE)
     RRA
     JP NC,TOOT  ;Ret via TOOT if ENTER pressed
     LD A,FE     ;Read keys SPACE ZXCV
     IN A,(FE)
     RRA
     RRA
     JR NC,NMCP  ;Jump if "Z" pressed
     RRA
     RRA
     JR C,NMSV   ;Scan again if C not pressed

;A warm reset to Basic via NMI keypress "C"
WARM LD A,(T/SP) ;Get Timex/Spectrum flag
     AND A       ;Timex?
     JR Z,TIMX   ;Jump if Timex mode
     LD HL,(rmtp) ;Get top of ram pointer
     LD (HL),3E  ;3EH at top of stack
     DEC HL      ;Point below 3E
     LD SP,HL    ;Stack starts below 3E
     DEC HL      ;Point 1 word below SP
     DEC HL
     LD (ersp),HL ;Error SP goes here
     LD HL,1303
CNTN PUSH HL
     LD IY,ernr  ;Point IY for Basic
     IM 1        ;Insure mode 1 interrupts
     LD L,FF     ;Set for error 0
     JP ER_L     ;Goto error#L+1 routine
TIMX LD A,(vmod) ;Get video mode flag
     LD HL,6200  ;Assume mode 0 video
     AND A       ;Is it mode 0?
     JR ZERO     ;Jump if it is
     LD HL,F9C0  ;Not mode zero so F9C0
ZERO LD (5CC0),HL
     DEC HL
     LD (HL),3E
     DEC HL
     LD SP,HL
     DEC HL
     DEC HL
     LD (ersp),HL
```

```
     LD HL,0E8D
     JR CNTN

;NMI COPY/ handler
NMCP LD A,(COPF) ;Get COPY/ flag
     AND A       ;Test
     JP Z,ACOP   ;ASCII copy if 0
     DEC A
     JP Z,OKCP   ;Okidata copy if 1
     DEC A
     JP Z,OLCP   ;Olivetti copy if 2
     DEC A
     JP Z,GMCP   ;Gemini/Epson copy if 3

;Gorilla Banana COPY/ routine
GBCP LD A,08     ;Lprint 08
     CALL PRTA
     LD BC,BF00
     LD H,1C
GLP1 LD A,1B     ;Lprint 1BH
     CALL PRTA
     LD A,10     ;Lprint 10H
     CALL PRTA
     XOR A       ;Lprint 00
     CALL PRTA
     LD A,70     ;Lprint 70H
     CALL PRTA
     LD L,B
GLP2 LD B,L
     LD D,07
GLP3 PUSH HL
     PUSH AF
     PUSH BC
     LD A,BF
     SUB B
     CCF
     JR NC,GJP1
     CALL PXAD
     LD B,A
     INC B
     LD A,(HL)
GLP4 RLCA
     DJNZ GLP4
     RRA
GJP1 RL H
     POP BC
     POP AF
     RR H
     POP HL
```

```
     RRA
     DEC B
     DEC D
     JR NZ,GLP3
     SCF         ;Set bit 7
     RRA
     CALL PRTA   ;Send dot pattern
     INC C
     JR NZ,GLP2
     LD A,0D     ;Lprint CR
     CALL PRTA
     DEC H
     JR NZ,GLP1
     LD A,0F     ;Lprint 0FH
     JP PRTA     ;Ret via lprintA routine

;Get last entry on calc stack
UNSK LD HL,(stnd)
     DEC HL
     LD B,(HL)
     DEC HL
     LD C,(HL)
     DEC HL
     LD D,(HL)
     DEC HL
     LD E,(HL)
     DEC HL
     LD A,(HL)
     LD (stnd),HL
SYNR LD A,(flgs)
     RLA
     RLA
     RET

;MODE 2 interrupt vector points here
ORG 0F0F
MD2I INC A       ;Mode 2 interrupt will cause A to inc
     RET         ;Ret from interrupt

;RESTORE fn TO fn handler
RX/X  RST 10H    ;Advance to next valid basic char
      CALL RSTK  ;Reset SP
      CALL REST/S ;Test RESTORE /S. No ret here if it is
ERC3  JP C,ER_C  ;Expression evaluated. Error if numeric
      CALL NWDN  ;Move name & type to TCAT
      CP CC      ;"TO" token?
ERC2  JP NZ,ER_C ;Error C if not
      RST 10H    ;Advance to next valid char
      CALL NXEX  ;Eval next expression
```

```
       JR C,ERC3  ;Error if numeric
       CALL SYN?  ;In syntax time?
       JR NC,SYPTH ;Skip if we are
       PUSH DE    ;Save pointer
       PUSH BC    ;And length
       CALL LCAT  ;Load the CAT to Bram buffer
       LD DE,TCAT ;Point at TCAT build-up area
       LD HL,CTFL ;Point at CATalog in Bram
       CALL MTCH  ;Try & find a match
       JP NC,ER_S ;Error S if not found
       LD (FLAD),HL ;Else store address of matching file
       POP BC     ;Restore new name length
       POP DE     ;Restore pointer to new name
       CALL GTST  ;Move new name to TCAT
       LD A,(DE)
       AND A
       JR NZ,ERC2
SYPTH  CALL CKND  ;Insure CR or ":" end statement
       CALL SYRT  ;Ret to rom in syntax time
       LD DE,(FLAD) ;Get address of filename to change
       LD HL,TCAT ;Point at new filename
       LD BC,000A ;Each name is 10d chars long
       LDIR       ;Insert new name into CAT
SC&R   CALL SCAT  ;Save the revised CAT
       JP DONE    ;Ret to rom in common code

PTCO   LD A,30
       AND A
NMLP   SBC HL,DE
       JR C,FINI
       INC A
       JR NMLP
FINI   ADD HL,DE
       CP 30
       JR Z,PCSP
       CALL PNTA
       LD A,30
       LD (PFLG),A
       RET
PCSP   LD A,(PFLG) ;Get print space/zero/nothing flag
       AND A       ;Zero?
       RET Z       ;Print nothing if zero
       JP PNTA     ;Else print space or zero

MV/H   RST 10H     ;Advance to next char
       CALL RSTK   ;Reset SP
       CP 0D       ;CR?
       JR Z,MVAL   ;Skip if it is
       CP 3A       ;Semicolon?
```

```
     JP NZ,MTO?  ;Jump to check not using defaults
MVAL CALL SYRT   ;Return in syntax time
     POP HL      ;Clear 1 word from stack
     CALL RCNT   ;Reset retry counter
     CALL RSR0   ;Restore current drive
     XOR A       ;A=00
     LD (TRK#),A ;Current track# is zero
     LD A,(DSEL) ;Get shadow drive/side select for this drive
     LD (DSL'),A ;Save it
     LD A,(DNUM) ;Get current drive#
     PUSH AF     ;Save it (SRC drive)
     INC A       ;Dest drive is next logical drive
     AND 03      ;Insure only bits 0 & 1 are affected
     LD (DNUM),A ;Make dest drive current drive#
     CALL FND#   ;Form DSEL from dest drive#
     POP AF      ;Restore SRC drive#
     LD (DNUM),A ;Make SRC drive current drive# again
     LD A,(DSEL) ;Get dest drive shadow select
     LD (DSL2),A ;Save dest drive select shadow
     CALL RSR2   ;Restore dest drive, too
GTLP CALL REDY   ;Insure controller ready
     LD A,(DSL') ;Get src drive select
     LD (DSEL),A ;Make it current drive sel
     OUT (B7),A  ;In the hardware, too
     CALL L5KB   ;Load 5K bytes from src drive
     JR C,OK__   ;Jump if load OK
     CALL RTRY   ;Make 1 bell tink & return if retrys left
     JR GTLP     ;Try & load save cyl again
OK__ CALL RCNT   ;Reset retry counter
ROUN LD A,(DSL2) ;Get dest drive shadow sel
     LD (DSEL),A ;Dest drive is now current drive
     OUT (B7),A  ;In hardware, too
     CALL S5KB   ;Save the cylinder on dest drive
     JR C,OK_2   ;Jump if save OK
RSV? LD A,(DSL') ;Select src drive in case no retrys left
     LD (DSEL),A
     OUT (B7),A
     CALL RTRY   ;Make a bell tink & ret here if retrys left
     JR ROUN     ;Try & save cylinder again
OK_2 LD A,(DSL') ;Select src drive again
     LD (DSEL),A
     OUT (B7),A
     LD A,(SIDS) ;Get src drive # of sides
     DEC A       ;Test flag
     JR Z,NEXTR  ;Jump if only 1 side this disk
     LD HL,SID#  ;Point at current side# variable
     DEC (HL)    ;Test
     INC (HL)
     JR NZ,NEXTR ;Jump for next track if last not side 0
```

```
     LD (HL),A   ;Else this time will be side 1
     LD HL,DSL'  ;Point at src drive DSEL
     SET 7,(HL)  ;Set src DSEL for side 1
     INC HL      ;Point at dest drive DSEL (DSL2)
     SET 7,(HL)  ;Dest drive uses side 1, too
     JR GTLP     ;Loop to send side 1

;Advance to next track
NEXTR LD HL,TRK# ;Point at current track#
     INC (HL)    ;Next track
     LD A,(TRKS) ;Get max # of tracks
     CP (HL)     ;Done?
     JP Z,RETB   ;Exit if done
     LD HL,(DSL') ;Get both DSELs
     LD A,H      ;Adjust so both will step in
     OR L
     PUSH AF     ;Save DSEL code that will select both
     LD A,(HDSP) ;Get head step speed
     AND 03      ;Mask unused bits
     OR 50       ;Combine with step-in command code
     EX AF,AF'   ;Save code
     CALL REDY   ;Insure controller ready
     POP AF      ;Retrieve DSEL code
     OUT (B7),A  ;Enable both src & dest drives
     EX AF,AF'   ;Get command to step in
     OUT (8F),A  ;Send command
     LD HL,DSL'  ;Point at src drive DSEL
     RES 7,(HL)  ;Now using side 0
     INC HL      ;Now point at dest drive DSEL (DSL2)
     RES 7,(HL)  ;Side 0 on dest drive, too
     XOR A       ;A=00
     LD (SID#),A ;Current side# is side 0
     JP GTLP     ;Loop to move next cylinder

;Here to load a single cylinder
CYLL  SCF        ;Carry means this is a LOAD
      JR LDJP    ;Jump to common code
;Here to save a single cylinder
CYLS  AND A      ;No carry means this is a save
LDJP  LD A,05    ;A=5
      LD (TRY2),A ;Set for 5 retrys
      LD (OLHL),HL ;Save HL
      LD (OLDE),DE ;Save DE
      JR NC,SVJP ;Jump if saving
LAGN  CALL LDCY  ;Else load the cylinder
      RET C      ;Done if no errors
      CALL RETRY? ;Check if retry available
      JR LAGN    ;Loop for another try
```

```
SVJP  CALL SVCY  ;Save the cylinder
      JR NC,NTHR ;Check retry if available
      CALL VFCY1 ;Verify the cylinder
      RET C      ;Done if no errors
NTHR  CALL RETRY? ;Check if retry available
      JR SVJP    ;Loop for another try

RETRY? CALL STOT ;Ring 1 bell
      LD HL,TRY2 ;Point at alternate retry counter
      AND A      ;Reset carry to signal possible error
      DEC (HL)   ;Dec counter. Set Z flag if zero
      POP DE     ;Remove last calling address
      RET Z      ;Ret w/o carry to signal error
      PUSH DE    ;Calling address back to stack
      LD HL,(OLHL) ;Retrieve old HL
      LD DE,(OLDE) ;Retrieve old DE
      RET

;Advance to next available cylinder
NEXCY LD A,(DSDS)
     DEC A
     JR Z,_1SID
     LD A,(SID#)
     AND A
     JR NZ,SD_1
NXC2 LD A,FF
     LD (SID#),A
     LD A,(DSEL)
     SET 7,A
     LD (DSEL),A
     OUT (B7),A
     RET

SD_1 LD A,(DSEL)
     RES 7,A
     LD (DSEL),A
     OUT (B7),A
     XOR A
     LD (SID#),A
_1SID LD A,(HDSP)
     AND 03
     OR 50
     PUSH AF
     CALL REDY
     POP AF
     OUT (8F),A

;Pause about 28 ms
PS28  CALL REDY
```

```
P28M  PUSH BC
      LD BC,0EB0
DELA  DEC BC
      LD A,B
      OR C
      JR NZ,DELA
      POP BC
      RET

GTST  CALL SYN?
      JR NC,SPATH
      CALL STST
SPATH LD HL,0018
      PUSH HL
      RST 20H
      CP E4
      JR NZ,NXT1
      LD A,01
      LD (DE),A
      RST 10H
      CP 24
      RET NZ
      LD A,02
      JR STTP
NXT1  CP AF
      JR Z,CDFL
      CP AA
      JR NZ,NXT2
CDFL  LD A,03
      JR STTP
NXT2  CP BD
      JR NZ,NXT3
      LD A,04
      JR STTP
NXT3  CP B0
      JR Z,VALF
      PUSH AF
      XOR A
      LD (DE),A
      POP AF
      RET

VALF  LD A,05
STTP  LD (DE),A
      RST 10H
      RET

;Entry to EXROM
ORG 1101
```

```
GOEX  LD A,(0008)

;MOVE /FN handler
MTO?  CALL NXEX  ;Evaluate next expression
      JP C,ER_C  ;Error C if numeric
      LD A,(DNUM) ;Get present drive#
      INC A      ;Bump to next logical drive
      AND 03     ;Insure wrap from drive 3 to drive 0
      LD (DDV#),A ;Store assumed dest drive#
      CALL GTST  ;Get string to TCAT & advance to next char
      CP CC      ;Token 'TO'?
      JR NZ,DFLT ;Jump to use default if not
      RST 10H    ;Advance past 'TO' token
      CALL EVA#  ;Evaluate as a number, error if not
      LD A,B     ;MSB of number to A
      AND A      ;Number >255?
      JR NZ,ERW2 ;Errow W if it is
      LD A,C     ;LSB of number to A
      CP 04      ;Number >3?
ERW2  JP NC,ER_W ;Error W if it is
      LD (DDV#),A ;Store Supplied dest drive#
DFLT  CALL CKND  ;Insure CR or ':' end statement
      CALL SYRT  ;Ret to rom in syntax time
      CALL RCNT  ;Reset retry counter
AGIN  CALL MVFN  ;Move the file
      JP DONE    ;Finish in common code

;Reset retry counter
RCNT  LD A,03    ;Set for 3 retrys
      LD (TRYS),A ;Set the counter
      RET

;MOVE /fn
MVFN  LD A,(DNUM) ;Get current drive#
      LD HL,SDV# ;Point at src drive#
      LD (HL),A  ;Same as current drive#
      INC HL     ;Point at dest drive# (DDV#)
      CP (HL)    ;Src & dest drive the same?
      JP Z,ER_W  ;Error W if they are
      CALL LCAT  ;Load the CATalog
      LD HL,CTFL ;Point to start of CAT entrys
      LD DE,TCAT ;Point at file to match
      CALL MTCH  ;Try and find a match
      JP NC,ER_S ;Error S if 'File Not Found'
      LD DE,TCAT ;Point at start of TCAT buffer
      LD BC,0014 ;Each CAT entry is 20d bytes long
      LDIR       ;Move actual CAT entry to buffer
      LD A,(TCYL) ;Get # of cylinders in file
      LD (TCNT),A ;This will be # of cyls to move
```

```
       LD A,(TSID) ;Get starting side# of file
       LD (SRSD),A ;Save it as source file side#
       LD (SID#),A ;And current side#
       LD A,(TTRK) ;Get file starting track#
       LD (TRK#),A ;Now current track#
       CALL FND#   ;Set DSEL using (SID#) & (TRK#)
       CALL RSR2   ;Restore the drive to track 0
       CALL SEKT   ;Position source drive at start of file
       RET NZ      ;Done if error
       CALL SVSC   ;Save src drive settings
       LD A,(DSDS) ;Get # of sides on source disk
       LD (SSDS),A ;Save info
       LD A,(DDV#) ;Get destination drive#
       LD (DNUM),A ;Make dest drive current drive
       CALL LCAT   ;Load dest drive CAT
       LD A,(NXCY) ;Get track# of next available cylinder
       LD (DTK#),A ;Save dest track#
       LD A,(NXSD) ;Get side# of next available cylinder
       LD (DESD),A ;Save dest side#
       LD A,(DSDS) ;Get # of sides on dest disk
       LD (DSTS),A ;Save info
       LD HL,(FRCY) ;Get # of free cylinders on dest disk
       LD DE,(TCNT) ;Get size in cylinders of file to move
       LD D,00     ;Insure enough room is available on dest drive
       AND A       ;for file to be moved
       SBC HL,DE
       LD A,(SDV#) ;Get source drive#
       LD (DNUM),A ;Now current drive#
ERU2   JP C,ER_U   ;Error U if not enough room to move file
       LD HL,(NXCA) ;Get address of next catalog entry on dest disk
       LD DE,33E0  ;Test to insure enough room for additional entry
       EX DE,HL
       SBC HL,DE
       JR C,ERU2   ;Error U if no room for new entry in dest catalog
       LD A,(DDV#) ;Get dest drive#
       LD (DNUM),A ;Now current drive#
       LD A,(DTK#) ;Get dest track#
       LD (TRK#),A ;Now current track#
       LD A,(DESD) ;Get dest side#
       LD (SID#),A ;Now current side#
       CALL FND#   ;Set DSEL for dest drive
       CALL RSR2   ;Restore dest drive to track 0
       CALL SEKT   ;Position dest drive at start of new file
       RET NZ      ;Done if error positioning
       CALL SVDS   ;Save dest drive settings
MLOP   CALL SELS   ;Select src drive
       CALL L5KB   ;Load one cyl to buffer
       RET NC      ;Done if error
       LD A,(DSDS) ;Save # of sides variable
```

```
      EX AF,AF'
      LD A,(SSDS) ;Get src drive # of sides
      LD (DSDS),A ;Store src drive # of sides for NEXCY routine
      CALL NEXCY  ;Advance src drive to next available cylinder
      EX AF,AF'   ;Restore # of sides variable
      LD (DSDS),A
      CALL SVSC   ;Save src drive settings
      CALL SELD   ;Select dest drive
      CALL S5KB   ;Save the cylinder onto dest drive
      RET NC      ;Done if error
      LD A,(DSTS) ;Get # of sides on dest drive
      LD (DSDS),A ;Insure info is available for NEXCY routine
      CALL NEXCY  ;Advance dest drive to next available cylinder
      CALL SVDS   ;Save dest drive settings
      LD HL,TCNT  ;Point at cylinder counter
      DEC (HL)    ;Decrement count
      JP NZ,MLOP  ;Loop to move next cylinder if not done
      CALL LCAT   ;Load dest drive catalog
      LD HL,(NXCY) ;Get old next available cyl track# & side#
      LD (TTRK),HL ;Now pointer to file just moved
      LD A,(DTK#) ;Get dest track#
      LD (NXCY),A ;Now new next available cyl track#
      LD A,(DESD) ;Get dest side#
      LD (NXSD),A ;Now new next available cyl side#
      LD HL,(FRCY) ;Adjust # of available cylinders variable
      LD DE,(TCYL) ;to reflect space just used
      LD D,00
      AND A
      SBC HL,DE
      LD (FRCY),HL
      LD DE,(NXCA) ;Get old next avail cat entry address
      LD HL,TCAT   ;Point at new entry in TCAT
      LD BC,0014   ;20d bytes per entry
      LDIR         ;Move new file entry into catalog area
      EX DE,HL     ;Exchange pointers
      LD (HL),80   ;Mark end of cat area w/80h
      LD (NXCA),HL ;Update next avail cat entry address variable
      CALL SCAT    ;Save updated catalog to dest drive
      CALL SELS    ;Select original (src) drive
      SCF          ;Show no errors
      RET          ;Done

;Olivetti PR2300 COPY / data
OVDAT DB 1B,47,31,3B,33,32,3B,30,32,34,1B,5A

;Olivetti PR2300 COPY / routine
OLCP  LD B,0C    ;12 numbers to send
      LD HL,OVDAT ;Point at data
OLL1  LD A,(HL)  ;Get a byte
```

```
       INC HL     ;Bump
       CALL PRTA  ;Send the byte
       DJNZ OLL1  ;Loop till all 12 sent
       LD HL,4000 ;Start of display file
       LD E,03    ;3 fields to the display
OLL5   LD D,08    ;8 char lines per field
OLL4   PUSH HL    ;Save field start address
       LD C,08    ;8 scan lines
OLL3   LD B,20    ;32 char spaces per scan line
       PUSH HL    ;Save start of scan line
OLL2   LD A,(HL)  ;Get byte to send
       CALL PRTA  ;Send the byte
       INC HL     ;Bump
       DJNZ OLL2  ;Loop till 32 char spaces sent for line
       POP HL     ;Restore start of scan line
       INC H      ;Next scan line
       DEC C      ;Dec scan line count
       JR NZ,OLL3 ;Loop till entire char line sent
       DEC D      ;Dec char line count
       JR Z,OLJ1  ;Jump if done with a display field
       POP HL     ;Restore last field start address
       LD A,20    ;Set A for offset
       ADD A,L    ;Add offset
       LD L,A     ;Point at next char line
       JR OLL4    ;Send another char line
OLJ1   POP AF     ;Clear stack
       DEC E      ;Dec dfile area counter
       LD A,E     ;Save counter
       RET Z      ;Done if all 3 dfile areas sent
       LD HL,4800 ;HL=4800=2nd dfile area
       CP 02      ;2nd area?
       JR Z,OLL5  ;Send 2nd area if it is
       LD H,50    ;HL=5000=last dfile area
       JR OLL5    ;Loop to send last area

SC$C   LD HL,(chas)
       INC H
       LD A,C
       RRCA
       RRCA
       RRCA
       AND E0
       XOR B
       LD E,A
       LD A,C
       AND 18
       XOR 40
       LD D,A
       LD B,60
```

```
SCLP  PUSH BC
      PUSH DE
      PUSH HL
      LD A,(DE)
      CALL XRHL
      JR Z,MATCH
      INC A
      JR NZ,SCNX
      DEC A
MATCH LD C,A
      LD B,07
SCRW  INC D
      INC HL
      LD A,(DE)
      CALL XRHL
      XOR C
      JR NZ,SCNX
      DJNZ SCRW
      POP BC
      POP BC
      POP BC
      LD A,80
      SUB B
      LD C,01
      RET
SCNX  POP HL
      LD DE,0008
      ADD HL,DE
      POP DE
      POP BC
      DJNZ SCLP
      LD C,B
      RET

;Wait till drive ready or BREAK pressed
REDY  IN A,(8F)  ;Get controller status
      RRA        ;Busy bit to carry
      RET NC     ;Return if ready
      CALL TBRK  ;Test BREAK. Error D if pressed
      JR REDY    ;Loop till either controller ready or BREAK

;Load 5K bytes to B bank buffer
L5KB  LD HL,2000 ;Point at start of B bank buffer
LD5K  LD DE,1400 ;5K bytes to load
LDHD  LD A,01    ;Start with sector number one
      LD (SEC#),A ;Save as current sector#
LHD2  CALL PS28  ;Pause 28ms
LDLP  LD A,D
      SUB 14
```

```
      JP C,CYLL
      JR NZ,LD_NEXT
      OR E
      JP Z,CYLL
      XOR A
LD_NEXT LD D,A
      PUSH DE
      LD DE,1400
      CALL CYLL
      POP DE
      RET NC
      CALL NEXCY
      JR LDLP

BUS?  RRA          ;Busy bit to carry
      JR C,DLOD    ;Continue if controller still active
      EXX          ;Else return
      RET

;Load a single cylinder from disk
LDCY CALL PREX    ;Prepare exchange registers for overflow
     LD A,E       ;LSB to A
     AND A        ;Zero?
     JR Z,LCY2    ;OK if LSB zero
     INC D        ;Else adjust MSB
LCY2 CALL REDY    ;Insure controller is ready
     LD A,(SEC#)  ;Get current sector#
     OUT (AF),A   ;Send it to controller
     LD A,90      ;90H is read sector mult command
     CALL SEND8F  ;Send command to controller
     LD C,BF      ;Port BF
     LD B,E       ;LSB of count to B
GBYT IN A,(8F)    ;Get controller status
     RRA          ;Busy bit to carry
     RET NC       ;Ret NC to signal error if not busy
     RRA          ;A byte ready to be read?
     JR NC,GBYT   ;Loop if not
     INI          ;Get the byte
     JR NZ,GBYT   ;Loop till E bytes read
     DEC D        ;MSB decrement
     JR NZ,GBYT   ;Loop until all bytes read
     EXX          ;Get prepared exchange registers
     DJNZ LDON    ;B is set as flag. Done if it was a 1
DLOD IN A,(8F)    ;Else read status register
     BIT 1,A      ;Is a byte ready?
     JR Z,BUS?    ;Jump if not
     IN A,(BF)    ;Do a dummy read
     DEC DE       ;Dec dummy read counter
     LD A,D       ;Test done w/dummy read
```

```
     OR E
     JR NZ,DLOD   ;Loop till DE' bytes trashed
LDON EXX          ;Back to main registers
     IN A,(AF)    ;Read controller sector register
     LD C,A       ;Store sector# in C
ENDL IN A,(8F)    ;Read controller status
     RRA          ;Busy bit to carry
     RET NC       ;Ret NC to signal error if not busy
     IN A,(AF)    ;Read sector register
     CP C         ;A change from last time?
     JR Z,ENDL    ;Loop till there is a change
HALT: LD A,D0     ;D0H will force the controller to stop
     CALL SEND8F  ;Send command to stop controller
     LD A,01
     LD (SEC#),A  ;Next sector# will be sector 1
     IN A,(8F)    ;Read controller status one last time
     AND 5C       ;Check bits 2,3,4,&6 (0x0x xx00)
     RET NZ       ;Signal error w/carry if any set
     SCF          ;Else signal no errors
     RET

;Save a single cylinder
SVCY CALL PREX    ;Prepare exchange registers for write
     LD A,E
     AND A
     JR Z,SCY2
     INC D
SCY2 CALL REDY    ;Insure controller is ready
     LD A,(SEC#)  ;Get starting sector#
     OUT (AF),A   ;Load WD1770 sector register
     LD A,B0
     CALL SEND8F  ;Send command to controller
     LD C,BF
     LD B,E
SNDB LD E,(HL)
     INC HL
SND2 IN A,(8F)
     RRA
     RET NC
     RRA
     JR NC,SND2
     OUT (C),E
     DJNZ SNDB
     DEC D
     JR NZ,SNDB
     EXX
     DJNZ FINISH
SND3 IN A,(8F)
     BIT 1,A
```

```
     JR Z,BUSY
     OUT (C),B
     DEC DE
     LD A,D
     OR E
     JR NZ,SND3
FINISH EXX
     IN A,(AF)    ;Get controller current sector#
     LD C,A       ;Save it for later
STOP IN A,(8F)    ;Get controller status
     RRA          ;Busy bit to carry
     RET NC       ;Error if controller not busy
     IN A,(AF)    ;Get controller current sector#
     CP C         ;Starting on next sector yet?
     JR Z,STOP    ;Loop if not
     JR HALT      ;Done if it is
BUSY RRA
     JR C,SND3
     EXX
     RET

;Save 5K bytes from B bank buffer
S5KB LD HL,2000
SV5K LD DE,1400
SVHD LD A,01
     LD (SEC#),A
SHD2 CALL PS28
SVLP LD A,D
     SUB 14
     JP C,CYLS
     JR NZ,SNXT
     OR E
     JP Z,CYLS
     XOR A
SNXT LD D,A
     PUSH DE
     LD DE,1400
     CALL CYLS
     POP DE
     RET NC
     CALL NEXCY
     JR SVLP

;Clear the entire screen
CLS!: CALL CALLS/T ;Call Spectrum or Timex rom
     DW 0DAF      ;Spectrum rom cls
     DW 08EA      ;Timex rom cls
     RET
```

```
;Here to interpret possible CAT/N command
CATW? LD HL,0018  ;Get current Basic character
      PUSH HL     ;(Basic's RST18H)
      RST 20H
      CP 0D       ;CR?
      RET Z       ;No change if it is
      CP ':'      ;":"?
      RET Z       ;No change if ":" either
      CALL NXEX   ;Evaluate next expression
      JP NC,ER_C  ;Error C if not numeric
      CALL SYN?   ;Syntax time?
      RET NC      ;Skip rest if syntax time
      LD A,B      ;Get MSB of integer
      AND A       ;Zero?
      JP NZ,ER_B  ;Error B if integer >255
      LD A,C      ;Get LSB of integer
      LD (WIDECAT),A ;Store the number. 0 returns to normal CAT
      RET         ;Go do the CATalog

;COPY / handler
CPY/ RST 10H      ;Advance to char after "/"
     CALL RSTK    ;Reset stack
     CALL CKND    ;Insure a CR or ":" ends statement
     CALL SYRT    ;Ret to rom in syntax time
     POP HL       ;Clear one addr from stack
     LD C,4F      ;Load reg C with "O" char
     CALL SETP    ;Insure Oliger Printer Port selected
     CALL NMCP    ;Call NMI COPY
     POP HL       ;Clear last 3 addr from stack
     POP HL
     POP HL
     LD HL,1B76   ;1B76=STATEMENT RET in Spec rom
     LD A,(T/SP)  ;Get Timex/Spec flag
     AND A        ;Test it
     JR NZ,NOTM   ;Jump if Spec rom active
     LD HL,1AB9   ;Set STMT RET addr for Timex rom
NOTM EX (SP),HL   ;Replace last addr with STMT RET addr
     RST 18H      ;Jump to STMT RET in either rom

;Here to print message #C
PMSC LD HL,MESS   ;Point at message table
     PUSH DE      ;Save DE
     LD A,(WIDECAT) ;Get widecat flag
     AND A        ;Normal CAT?
     LD E,FF      ;Max message length is 255 chars
     JR Z,PNHL    ;Skip in normal CAT mode
     LD A,C       ;Get message#
     CP 01        ;Message#1?
     JR NZ,NEXTM  ;Skip if message not #1
```

```
     LD E,22      ;#1 message will be 34 characters in widecat
     JR PNHL      ;Go print message#1
NEXTM CP 10       ;Message# <16d?
     JR C,PNHL    ;Skip if message #2-#15d
     CP 16        ;Message# >21d?
     JR NC,PNHL   ;Skip if message #22 or greater
     LD E,03      ;Messages #16d-21d only 3 chars long now
PNHL LD A,(HL)    ;Get char
     INC HL       ;Point at next
     RLA          ;Bit 7 to carry
     JR NC,PNHL   ;Loop till end of mess via bit 7 set
     DEC C        ;Correct message?
     JR NZ,PNHL   ;Loop if not
PTLP LD A,(HL)    ;Get char
     RES 7,A      ;Insure end of message bit reset
     CALL PNTA    ;Print the char
     DEC E        ;Dec character count
     JR Z,MESSEND ;Done if all characters sent
     LD A,(HL)    ;Get the char again
     RLA          ;Last char?
     JR C,MESSEND ;Done if so
     INC HL       ;Next char
     JR PTLP      ;Loop to send it
MESSEND POP DE    ;Restore DE
     RET

NWDN LD HL,0018   ;Get present Basic character
     PUSH HL
     RST 20H
     CP 0D        ;CR?
     JR Z,NWD2    ;Skip if it is
     CP 3A
     JP NZ,GTST
NWD2 POP HL
     CALL SYRT
     PUSH DE
     PUSH BC
     CALL LCAT
     POP BC
     POP HL
     LD A,C
     CALL MDNA
     JP SC&R

;MERGE / command handler
H/MG LD HL,(prog) ;Get start of current Basic program
     LD (OPGM),HL ;Save pointer
     CALL SYN?    ;Syntax time?
     LD A,80      ;Signal MERGE in progress
```

```
     JR NC,SVCN   ;Skip in syntax time
     LD HL,(vars) ;Point sv prog past real prog area
     LD (prog),HL
     JR SVCN      ;Cont in common code

;Handle SAVE /
H/SA XOR A        ;Signal SAVE in progress
     JR SVCN      ;Cont in common code

;Handle LOAD/
H/LD LD A,FF      ;Signal LOAD in progress
SVCN LD (SV/L),A  ;Save SAVE/MERGE/LOAD/RUN flag (F7=RUN)
     CP 80        ;MERGE in progress?
     EX AF,AF'    ;Save merge flag (Z)
     RST 10H      ;Advance to char after first "/"
     LD (WFLG),A  ;Save this character
     CP 2F        ;Was it another "/"?
     JR NZ,NORM   ;Cont if not
     RST 10H      ;Advance past 2nd "/"
NORM CALL RSTK    ;Reset stack
     CALL NXEX    ;Eval next expression
     JP C,SL/N    ;Cont in SL/N if numeric
     CALL SYN?    ;Syntax time?
     JR NC,SYCT   ;Skip in syntax time
     CALL STST    ;Get filename, etc., into TCAT area
     XOR A        ;Assume Basic file type
     LD (DE),A    ;Store file type byte
SYCT LD HL,0018   ;Set to do RST18H in home bank
     PUSH HL      ;Address to stack
     RST 20H      ;Get current Basic character
     CP AF        ;CODE?
     JP Z,CODE
     LD C,A       ;Save type code
     LD A,(SV/L)  ;Get flag
     CP F7        ;RUN/?
     JR NZ,SYCT2  ;Skip if not RUN
     LD A,(SYNSAVE) ;Get Basic's syntax flag
     LD (flgs),A  ;Restore it
     JR ERRORC    ;Error C for RUN when not a CODE file
SYCT2 LD A,C       ;Restore type code
     CP 0D        ;CR?
     JR Z,PRGM    ;Basic if CR
     CP ':'       ;":"?
     JR Z,PRGM    ;Basic if ":"
     EX AF,AF'    ;Get MERGE flag
     JR Z,MG_C    ;Error C if anything but Basic
     EX AF,AF'    ;Restore type code
     CP CA        ;LINE?
     JR Z,LINE
```

```
     CP E4         ;DATA?
     JP Z,DATA
     CP AA         ;SCREEN$?
     JP Z,SCR$
     CP BD         ;ABS?
     JP Z,/ABS
     CP B0         ;VAL?
     JR NZ,ERRORC ;Error C if none of the above

;Handle Variables LOAD/SAVE
/VAL LD A,05       ;Variables file is type#5
     LD (DE),A     ;So show it
     RST 10H       ;Advance past VAL token
     CALL CKN2     ;Insure CR or ":" end statement
     CALL SYRT     ;Done in syntax time
     LD HL,(vars) ;Get pointer to variables
     LD (TBEG),HL ;Start pointer for this file
     EX DE,HL      ;Save pointer to start
     LD HL,(elin) ;Get end of variables pointer +1
     SCF           ;Set carry to kill +1
     SBC HL,DE     ;Form length of variables file
     LD (TLEN),HL ;Store length of file
DOIT CALL SL@N     ;Save or load
DONE JP NC,ER_T   ;Disk I/O error if no carry
DONEOK LD SP,(SPST) ;Reset SP
     POP HL        ;Back up one more word
     JP RETB       ;Ret to Basic rom w/o errors

;Here if error C on MERGE
MG_C LD HL,(OPGM) ;Get old start of Basic program pointer
     LD (prog),HL ;Restore pointer
ERRORC JP ER_C      ;Abort w/error C report

;Here to handle Basic's LINE save argument
LINE: LD A,(SV/L) ;Get type of file
     AND A         ;Basic file?
     JP NZ,ER_C   ;Error C if not
     RST 10H       ;Advance past LINE token
     CALL NXEX     ;Evaluate the expression
     JR NC,ERRORC ;Error C if argument not numeric
     CALL SYN?     ;Syntax time?
     JR NC,PGM2   ;Skip in syntax time
     LD HL,D8F0   ;Set for 9999 max value test
     ADD HL,BC     ;Test upper line# limit
     JP C,ER_B     ;Error B if >9999
     LD H,B        ;Else start line# to HL
     LD L,C
     JR PGM1
PRGM LD HL,4000   ;Mark file as non-auto run
```

```
PGM1 LD (TBEG),HL ;Store auto-start line#
PGM2 CALL CKND    ;Insure CR or ":" end statement
     CALL SYRT    ;Done in syntax time
     LD A,(SV/L)  ;Get SAVE/LOAD/MERGE/RUN sysvar
     AND A        ;Test flag
     JR NZ,DOIT   ;Go load file if SAVE not requested
     LD HL,(elin) ;Get end of Basic program+1
     LD DE,(prog) ;Get start of Basic program
     SCF          ;Set to compensate for end+1
     SBC HL,DE    ;Find length of program & variables
     LD (TLEN),HL ;Save length
     JR Z,DONEOK  ;Retw/o error report length=00
     LD HL,(vars) ;Get start of variables address
     AND A        ;No carry
     SBC HL,DE    ;Figure offset to variables
     LD (TOFS),HL ;Save variables offset
     JR DOIT      ;Go save the program

PGLM LD HL,(elin)
     LD DE,(prog)
     SCF
     SBC HL,DE
     PUSH DE
     EX DE,HL
     LD HL,(TLEN)
     SCF
     SBC HL,DE
     JR C,ENUF
     LD DE,0005
     ADD HL,DE
     LD B,H
     LD C,L
     CALL ROOM
ENUF POP DE
     LD HL,(elin)
     DEC HL
     EX AF,AF'
     JR NZ,NORMAL
     LD H,D
     LD L,E
NORMAL EX AF,AF'
     LD A,(T/SP)
     LD BC,19E5
     AND A
     JR NZ,SIN8
     LD BC,174D
SIN8 PUSH BC
     RST 20H
     LD BC,(TLEN)
```

```
     CALL CALLS/T ;Call Spectrum or Timex rom
     DW 1655      ;Spectrum insert routine
     DW 12BB      ;Timex insert routine
     INC HL
     LD BC,(TOFS)
     ADD HL,BC
     LD (vars),HL
     EX AF,AF'
     JR Z,MEND    ;Jump if MERGE in effect
     EX AF,AF'
     LD HL,(TBEG)
     LD (nwpc),HL
     XOR A
     LD (nspc),A
     JP S/L9

;Here at end of MERGE
MEND CALL S/L9    ;Merge program
RPRG LD HL,(OPGM) ;Get back start of main program
     LD (prog),HL ;Reset pointer correctly
     RET

SCR$ RST 10H
     LD HL,4000
     LD (TBEG),HL
     LD BC,1B00
     XOR A
     LD (CODF),A
     JP COD2

/ABS LD A,04
     LD (DE),A
     RST 10H
     CALL CKN2
     CALL SYRT
     LD HL,C200
     LD (TLEN),HL
     LD HL,3E00
     LD (TBEG),HL
     LD SP,37FF
     EXX
     LD (HL'S),HL
     EXX
     XOR A
     LD (NMIF),A
     LD A,(SV/L)
     AND A
     JP Z,DOIT
     CALL SL@N
```

```
      JP NC,RDER
      LD A,(NMIF)
      AND A
      JP NZ,NMSR
      JP DONEOK

VRLM LD HL,(TLEN)
      PUSH HL
      LD DE,0005
      ADD HL,DE
      LD B,H
      LD C,L
      CALL CALLS/T ;Call Spectrum or Timex rom
      DW 1F05      ;Spectrum memtest routine
      DW 1FBB      ;Timex memtest routine
      POP BC
      LD HL,(elin)
      DEC HL
      CALL CALLS/T ;Call Spectrum or Timex rom
      DW 1655      ;Spectrum insert routine
      DW 12BB      ;Timex insert routine
      JP DMC3      ;Continue in common code

RUN/ LD HL,3800   ;Point at RUN/ buffer
      LD (TBEG),HL ;Store pointer
      LD HL,06E0   ;Buffer is from 3800-3EDF (06E0 bytes)
      LD (TLEN),HL ;Set max file length
      LD A,03      ;Bytes file is type#3
      LD (TTYP),A  ;Store type#
      CALL SL@N    ;LOAD the code to the buffer
      PUSH AF      ;Save flags
      LD A,(SYNSAVE) ;Get Basic's syntax flag
      LD (flgs),A  ;Restore the flag
      POP AF       ;Restore flag register
      JP NC,ER_T   ;Stop w/error T if read error
      LD A,FF      ;Set A=FFH
      LD (GSFLAG),A ;Signal RUN/ code loaded
      CALL 3800    ;Call routine
      CALL ENDLINE ;Skip all characters till CR or ":"
      JP DONEOK    ;Return w/o errors

;Here to SAVE/LOAD a CODE (BYTES) file (also run)
CODE EX AF,AF'    ;Get Merge flag
      JP Z,MG_C    ;Error if Merge command
      EX AF,AF'
      XOR A        ;Assume not using default start & length
      LD (CODF),A
      LD A,(SV/L)  ;Get SAVE/LOAD/MERGE/RUN flag
      CP F7        ;Run command?
```

```
     JR Z,RUN/    ;Handle RUN/ if in effect
     RST 10H      ;Advance to next Basic char
     CP 0D        ;A CR?
     JR Z,CR_OK   ;Jump if a CR
     CP 3A        ;Is it ":"?
     JR NZ,GNUM   ;Go to get number if not CR or ":"
CR_OK LD A,03     ;Show we are using default start & length
     LD (CODF),A
     LD A,(SV/L)  ;But are we to do a SAVE?
     AND A
     JR NZ,COD2   ;Continue later if not SAVE
ERC_3 JP ER_C     ;Error C. Nonsence in Basic
GNUM CALL NXEX    ;Evaluate next expression
     JR NC,ERC_3  ;Error C if not numeric
     LD (TBEG),BC ;Store start address
     LD DE,0018   ;Get present Basic character
     PUSH DE
     RST 20H
     CP 2C        ;Is it ","?
     JR Z,NEXT1   ;Go get length if it is
     CP 0D        ;Is it a CR or ":" character?
     JR Z,CR.OK
     CP 3A
     JR NZ,ERC_3  ;Error C if not
CR.OK LD A,(SV/L) ;Are we doing a SAVE?
     AND A
     JR Z,ERC_3   ;Error 3 if we are. Need length!
     LD A,02      ;Show we are using default length
     LD (CODF),A
     JR COD2      ;Skip trying to get a length
NEXT1 RST 10H     ;Advance to character after ","
     CALL NXEX    ;Evaluate next expression
     JR NC,ERC_3  ;Error C if not numeric
COD2 LD (TLEN),BC ;Store length of code block
     LD A,03      ;A type code of 3 is a bytes file
     LD (TTYP),A  ;Store the file type
S/LC CALL CKND    ;Insure CR or ":" end statement
     CALL SYRT    ;Ret to rom in syntax time
     JP DOIT      ;Go SAVE or LOAD the BYTES file

;Here to SAVE/LOAD a DATA file
DATA XOR A        ;Assume array already exists w/this name
     LD (CODF),A
     RST 10H      ;Advance to next Basic character
     CALL CALLS/T ;CALL Spectrum or Timex rom
     DW 28B2      ;Spectrum look_vars routine
     DW 2C70      ;Timex look_vars routine
     SET 7,C      ;Insure an array IS the type
     JR NC,OLDA   ;Jump if array did exist
```

```
     LD A,FF      ;Show that a new array must be created
     LD (CODF),A
     LD A,(SV/L)  ;Get SAVE/LOAD/MERGE flag
     AND A        ;Test flag
     JR NZ,OKLD   ;OK if LOAD was specified

;Error 2
ER_2 LD L,01      ;Stop w/error 2 because a SAVE DATA was
     JP ER_L      ;Requested and the array doesn't exist

OLDA JR NZ,ERC_3  ;Error C if not an array or string variable
     CALL SYN?    ;Are we in syntax time?
     JR NC,SNCT   ;Jump in syntax time
     LD A,(HL)    ;Get array type
     RLA          ;Test type
     JP NC,ER_3   ;Error 3 if not an array
     INC HL       ;Point at LSB of array length
     LD E,(HL)    ;Get LSB
     INC HL       ;Point to MSB of length
     LD D,(HL)    ;Get MSB
     LD (TLEN),DE ;Save length of array
     LD (BCST),DE ;Twice
     INC HL       ;Point to start of array
     LD (TBEG),HL ;Save start address
     LD (DEST),HL ;Twice
OKLD LD A,C       ;Save array type and name
     LD (AFST),A
     LD A,01      ;Assume a numerical array
     BIT 6,C      ;Is it numerical?
     JR Z,NUME    ;Jump if it is
     INC A        ;Show a character array
NUME LD (TTYP),A  ;Store type code
SNCT RST 10H      ;Get next Basic character
     CP 29        ;Next character ")"?
     JR NZ,OLDA   ;Error C if not
     RST 10H      ;Advance to next character after ")"
     JR S/LC      ;Continue within main SAVE/LOAD routine

DTLM LD HL,(TLEN)
     LD A,(CODF)
     AND A
     JR Z,OLD_DE
     INC HL
     INC HL
     INC HL
     JR DMCN
OLD_DE LD DE,(BCST)
     SCF
     SBC HL,DE
```

```
     JR C,EROO
DMCN LD DE,0005
     ADD HL,DE
     LD B,H
     LD C,L
     CALL CALLS/T ;Call Spectrum or Timex rom
     DW 1F05      ;Spectrum memtest routine
     DW 1FBB      ;Timex memtest routine
EROO LD HL,(TLEN)
     LD A,(CODF)
     AND A
     JR NZ,DMC2
     LD BC,(BCST)
     LD HL,(DEST)
     DEC HL
     DEC HL
     DEC HL
     INC BC
     INC BC
     INC BC
     CALL CALLS/T ;Call Spectrum or Timex rom
     DW 19E8      ;Spectrum reclaim2 routine
     DW 1750      ;Timex reclaim2 routine
DMC2 LD HL,(elin)
     DEC HL
     LD BC,(TLEN)
     INC BC
     INC BC
     INC BC
     CALL CALLS/T ;Call Spectrum or Timex rom
     DW 1655      ;Spectrum insert routine
     DW 12BB      ;Timex insert routine
     INC HL
     LD A,(AFST)
     LD (HL),A
     LD DE,(TLEN)
     INC HL
     LD (HL),E
     INC HL
     LD (HL),D
DMC3 INC HL
     LD (TBEG),HL
     JP S/L9

;Message table...
MESS DB 80        ;Message table starts with 80H
     DB 'JLO SAFE V2.65  ',7F,'1993, J. Oliger',0D,0D
     DB 'DISK NAME:',A0
     DB 0D,'FORMATTED ',C0
```

```
      DB ' TRACKS,',A0
      DB ' SIDE(S)',0D,'CAPACITY:',A0,FF
      DB 'K BYTES',8D
      DB 'FREE:',A0,FF
      DB 'TOTAL FILES:',A0,FF,FF
      DB 'S File not found,',A0
      DB 'T Disk I/O error,',A0
      DB 'U Disk full,',A0
      DB 'FILE EXISTS!! 5 seconds to abor',F4
      DB 'BASI',C3
      DB 'N AR',D2
      DB 'C AR',D2
      DB 'BYTE',D3
      DB 'STAT',C5
      DB 'VRBL',D3
      DB ' FILENAME  TYPE CYLS SIZE START',8D
      DB 'CYLS',AF
      DB 'V File too large,',A0
      DB 'W Invalid drive#,',A0
      DB 'X FOR/ w/o LET,',A0
      DB 'Y fp FOR/ variable,',A0
      DB 'Z neg FOR/ variable,',A0

;CATALOG routine. If (WIDECAT)=0 then normal CAT
;If >0 & <256 then do wideCAT with this many colunms
ORG 1906
CATL CALL LCAT    ;Load catalog into B ram buffer
CAT2 CALL CLS!    ;Clear screen
     LD A,02      ;Now open channel #2
     CALL CALLS/T ;Call Spectrum or Timex rom
     DW 1601      ;Spectrum open channel A routine
     DW 1230      ;Timex open channel A routine
     LD A,(flgs)  ;Insure copyright sign will print as such
     PUSH AF
     SET 4,A
     LD (flgs),A
     LD C,01      ;Print top line of catalog (message #1)
     CALL PMSC    ;AND "DISK NAME; "
     POP AF       ;Restore rom flag
     LD (flgs),A
     LD A,(WIDECAT) ;Get wideCAT flag
     INC A        ;Extra column on first print
     LD D,A       ;Set column counter
     DEC A        ;Back to zero if normal CAT
     AND A        ;WideCAT?
     JR NZ,WIDE1  ;Skip in widecat mode
     LD B,10      ;16d characters in disk name
     LD HL,DNAM   ;Point at disk name
NLOP LD A,(HL)    ;Print the name
```

```
     CALL PNTA
     INC HL
     DJNZ NLOP
     LD C,02       ;Print CR &"FORMATTED @" (message #2)
     CALL PMSC
     LD HL,(DTKS) ;Get # of tracks disk contains
     CALL PNTL     ;Print the number
     LD C,03       ;Print " TRACKS, " (message #3)
     CALL PMSC
     LD HL,(DSDS) ;Get # of sides on disk
     CALL PNTL     ;Print the number
     LD C,04       ;Print " SIDE(S)" CR (message #4)
     CALL PMSC     ;& "CAPACITY: "
     LD HL,(MXCY) ;Get max number of cylinders this disk
     CALL PNTL     ;Print it
     CALL SPAC     ;Print a space
     LD C,17       ;Print " CYLS/" (message #17H)
     CALL PMSC
     LD A,(MXCY)  ;Get max number of cylinders again
     CALL D&PA     ;Convert this to bytes and print it
     LD C,06       ;Print "K BYTES" CR (message #6)
     CALL PMSC
     LD C,07       ;Print "FREE: "
     CALL PMSC
     LD HL,(FRCY) ;Get number of cylinders free
     CALL PNTL     ;Print number of cylinders free
     CALL SPAC     ;Print a space
     LD C,17       ;Print " CYLS/" (message #17H)
     CALL PMSC
     LD A,(FRCY)  ;Get number of free cylinders again
     CALL D&PA     ;Convert to bytes free and print the no.
     LD C,06       ;Print "K BYTES" CR (message #6)
     CALL PMSC
     CALL CRET     ;Print another CR (skip a line)
     LD C,16       ;Print "FILENAME.........START" (msg #16)
     CALL PMSC
     LD B,20       ;Print 32d "-" characters
LNLP LD A,2D
     CALL PNTA
     DJNZ LNLP
     CALL CRET     ;Send a CR
WIDE1 LD HL,CTFL  ;Point at CATalog file
     XOR A         ;A=00
     LD (NMIF),A  ;Start with zero files
     JR BEGIN      ;Start at end of CAT loop
FLLP LD A,(NMIF)  ;Get file counter
     INC A         ;Bump
     LD (NMIF),A  ;Save file counter
     LD B,0A       ;10d chars per filename
```

```
     LD A,(WIDECAT) ;Get widecat flag
     AND A          ;Widecatalog?
     JR Z,NAMELP    ;Jump if not widecat
     DEC D          ;Dec column counter
     JR NZ,NAMELP   ;Skip if no CR yet
     LD D,A         ;Preset to maxcol#
     CALL CRET      ;Send a CR
NAMELP LD A,(HL)    ;Get a character
     CALL PNTA      ;Print it
     INC HL         ;Bump
     DJNZ NAMELP    ;Loop till 10 sent
     CALL SPAC      ;Print a space
     LD A,(HL)      ;Get filetype byte
     PUSH AF        ;Save it
     INC HL         ;Bump
     PUSH HL        ;Save pointer
     ADD A,10       ;Convert filetype byte to its message#
     LD C,A
     CALL PMSC      ;Print Filetype message
     CALL SPAC      ;Print a space
     POP HL         ;Restore pointer
     LD A,(WIDECAT) ;Get widecat flag
     AND A          ;Widecat in effect?
     JR Z,NOTWIDE   ;Skip in normal CAT mode
     POP AF         ;Clear filetype byte from stack
     CALL SPAC      ;Print another space
     LD BC,0009     ;Skip next 9 bytes
     ADD HL,BC      ;Form pointer to next CAT entry
     JR BEGIN       ;Skip normal CAT portion
NOTWIDE LD E,(HL)
     INC HL
     LD D,(HL)
     PUSH DE
     INC HL
     LD E,(HL)
     INC HL
     LD D,(HL)
     EX DE,HL
     EX (SP),HL
     PUSH HL
     LD HL,0005
     ADD HL,DE
     LD A,(HL)
     PUSH HL
     LD L,A
     LD H,00
     CALL JTEN
     CALL SPAC
     POP HL
```

```
     EX (SP),HL
     CALL PJHL
     CALL SPAC
     POP HL
     EX (SP),HL
     EX DE,HL
     POP HL
     EX (SP),HL
     LD A,H
     CP 03
     CCF
     JR Z,SSTR
     AND A
     JR NZ,CONTIN
     LD A,D
     CP 30
SSTR EX DE,HL
     CALL C,PJHL
CONTIN CALL CRET  ;Send a CR
     POP HL       ;Restore pointer
     INC HL       ;Bump to start of next file
BEGIN LD A,(HL)   ;Get first char
     CP 80        ;End of file?
     JR NZ,FLLP   ;Jump if not end of file
EMPT CALL CRET    ;Send a CR
     LD C,09      ;Print "TOTAL FILES: " message
     CALL PMSC
     LD HL,(NMIF) ;Get total # of files in catalog
     CALL PNTL    ;Print the number

;Send a CR to screen
CRET LD A,0D      ;Set for CR
     JP PNTA      ;Print a CR

;Error X
ER_X LD A,05
     JR NERR

;Here if overwrite found
OVWR LD A,(WFLG)  ;Get warning message flag
     CP 2F        ;Is it the "/" character?
     RET Z        ;Skip overwrite warning & delay if it is
     PUSH HL      ;Save address
     XOR A        ;Assume screen will not need clearing
     LD (STAF),A
     LD A,(TTYP)  ;Get file type
     CP 03        ;BYTES file?
     JR Z,CONTINU ;Skip message on BYTES files
     CP 04        ;STATE file?
```

```
     JR Z,CONTINU ;Skip message for STATE files too
     LD A,FF      ;Show that lower screen will need cleared
     LD (STAF),A
     CALL CLOW    ;Clear the lower screen
     LD C,0F      ;Print message #0FH
     CALL PMSC    ;"FILE EXISTS! 5 SECONDS TO ABORT"
CONTINU CALL TOOT ;Sound 7 bell tinks to alert user
     LD B,00      ;Wait about 5 seconds or so to give user
_60TH EI          ;a chance to abort via the BREAK key
     HALT
     CALL TBRK
     LD A,BF
     IN A,(FE)
     RRA
     JR NC,DO_IT  ;But abort wait if ENTER key pressed
     DJNZ _60TH   ;Loop till timer times out
DO_IT LD A,(STAF) ;Get CLS lower screen flag
     AND A        ;Test it
     CALL NZ,CLOW ;Clear lower screen if indicated
     POP HL       ;Restore address
     RET

;Error S
ER_S LD A,(SV/L)  ;Merge in progress?
     CP 80
     CALL Z,RPRG  ;Reset old pgm if it was a merge
     XOR A        ;Show error S (first new error)

;New error handler
NERR LD SP,(SPST) ;Reset stack pointer
     PUSH AF      ;Save error code
     LD A,(T/SP)  ;Timex or Spectrum rom?
     AND A
     JR NZ,ERCN   ;Continue further on if Spectrum
     BIT 7,(IY+7D) ;Is ON ERR in effect?
     JR Z,ERCN    ;Continue later if not
     SET 6,(IY+7D) ;Show that an error occurred
     POP AF       ;Restore error code
     ADD A,1C     ;Add offset for new errors
     LD (errt),A  ;Store the error code
     LD HL,0EA3   ;Want to jump to add 0EA3 in Timex rom
     EX (SP),HL   ;Put address on stack
     RST 18H      ;And jump to it
ERCN POP AF       ;Restore error code
     ADD A,0C     ;Add offset for error text look-up
     CP 0F        ;Is adjusted code <0F?
     JR C,FIRST   ;Leave alone if it is
     ADD A,09     ;Adjust to access error from 2nd block
FIRST PUSH AF     ;Save error message look-up code
```

```
     XOR A         ;Let A=00
     LD H,A        ;Clear HL
     LD L,A
     LD (IY+37),H ;Clear several rom system variables
     LD (IY+26),H
     LD (5C0B),HL ;Clear dfad
     INC HL
     LD (5C16),HL
     CALL CALLS/T ;Call Spectrum or Timex rom
     DW 16B0      ;Spectrum set_min routine
     DW 133F      ;Timex set_min routine
     LD A,07      ;Insure sound chip is off
     OUT (F5),A
     LD A,FF
     OUT (F6),A
     RES 5,(IY+1)  ;Reset bit 5 of sv flgs
     RES 3,(IY+2)  ;Reset bit 3 of tvfl
     CALL CLOW     ;Clear lower portion of display
     POP AF        ;Restore new error message look-up code
     LD C,A        ;Move the code to C
     CALL PMSC     ;Print message #C
     LD SP,(ersp)  ;Put the stack where the rom expects it
     SET 5,(IY+2)  ;Set bit 5 of tvfl
     LD HL,1350    ;Want to continue at 1350 in Spec rom
     LD A,(T/SP)   ;Timex or Spec rom?
     AND A
     JR NZ,SINCL2  ;Jump if Spec rom
     LD HL,0F20    ;Else continue at 0F20 in Timex rom
SINCL2 EX (SP),HL  ;Put address onto stack
     RST 18H       ;And jump to it

;Error Z
ER_Z LD A,07        ;7th new error message
     JR NERR        ;Jump to new error handler

;Error Y
ER_Y LD A,06        ;6th new error message
     JR NERR        ;Jump to new error handler

;Mode 2 int vector of 0F must stay here as indicated
ORG 1B00          ;Start of mode 2 vector table
NMVC DB 0F        ;All 0FH must stay for mode 2 vector
ROOM LD HL,(stnd) ;1B01-1B03
     DB 0F,0F,0F,0F,0F ;1B04-1B08
     ADD HL,BC    ;1B09
     JR C,ER4X    ;1B0A-1B0B
     DB 0F,0F,0F,0F,0F ;1B0C-1B10
     EX DE,HL     ;1B11
     XOR A        ;1B12
```

```
     AND A         ;1B13
     DB 0F,0F,0F,0F,0F ;1B14-1B18
     LD HL,0050    ;1B19-1B1B
     DB 0F,0F,0F,0F,0F ;1B1C-1B20
     ADD HL,DE     ;1B21
     JR C,SPEC1    ;1B22-1B23
     DB 0F,0F,0F,0F,0F ;1B24-1B28
     EX DE,HL      ;1B29
     DB 0F,0F,0F,0F,0F,0F,0F ;1B2A-1B30
     LD HL,T/SP    ;1B31-1B33
     DB 0F,0F,0F,0F,0F ;1B34-1B38
     CP (HL)       ;1B39
     JR Z,TIMEX    ;1B3A-1B3B
     DB 0F,0F,0F,0F,0F ;1B3C-1B40
     LD HL,0000    ;1B41-1B43
     DB 0F,0F,0F,0F,0F ;1B44-1B48
     ADD HL,SP     ;1B49
     JR SPEC1      ;1B4A-1B4B
     DB 0F,0F,0F,0F,0F ;1B4C-1B50
TIMEX LD HL,(rmtp) ;1B51-1B53
     DB 0F,0F,0F,0F,0F ;1B54-1B58
SPEC1 EX DE,HL     ;1B59
     DB 0F,0F,0F,0F,0F,0F,0F ;1B5A-1B60
     SBC HL,DE     ;1B61-1B62
     RET C         ;1B63
     DB 0F,0F,0F,0F,0F ;1B64-1B68
ER4X LD HL,(SV/L) ;1B69-1B6B
     DB 0F,0F,0F,0F,0F ;1B6C-1B70
     BIT 7,L       ;1B71-1B72
     DB 0F,0F,0F,0F,0F,0F ;1B73-1B78
     JR Z,ER_4     ;1B79-1B7A
     DB 0F,0F,0F,0F,0F,0F ;1B7B-1B80
     BIT 6,L       ;1B81-1B82
     DB 0F,0F,0F,0F,0F,0F ;1B83-1B88
     JR NZ,ER_4    ;1B89-1B8A
     DB 0F,0F,0F,0F,0F,0F ;1B8B-1B90
     LD HL,(OPGM) ;1B91-1B93
     DB 0F,0F,0F,0F,0F ;1B94-1B98
     LD (prog),HL ;1B99-1B9B
     DB 0F,0F,0F,0F,0F ;1B9C-1BA0
ER_4 LD L,03       ;1BA1-1BA2
     DB 0F,0F,0F,0F,0F,0F ;1BA3-1BA8
     JP ENTL       ;1BA9-1BAB
     DB 0F,0F,0F,0F,0F ;1BAC-1BB0
CAT/ RST 10        ;1BB1     ;Adv to next char
     DB 0F,0F      ;1BB2-1BB3
     DB 0F,0F,0F,0F,0F ;1BB4-1BB8
     CALL CATW?    ;1BB9-1BBB ;Handle possible WideCAT change
     DB 0F,0F,0F,0F,0F ;1BBC-1BC0
```

```
HCAT CALL RSTK    ;1BC1-1BC3 ;Reset stack pointer
     DB 0F,0F,0F,0F,0F ;1BC4-1BC8
     CALL CKND    ;1BC9-1BCB ;Insure CR or ":" end statement
     DB 0F,0F,0F,0F,0F ;1BCC-1BD0
     CALL SYRT    ;1BD1-1BD3 ;Ret to rom in syntax time
     DB 0F,0F,0F,0F,0F ;1BD4-1BD8
     CALL CATL    ;1BD9-1BDB ;Perform the CATalog
     DB 0F,0F,0F,0F,0F ;1BDC-1BE0
     POP HL       ;1BE1 ;Adjust stack pointer
     DB 0F,0F     ;1BE2-1BE3
     DB 0F,0F,0F,0F,0F ;1BE4-1BE8
     JP RETB      ;1BE9-1BEB ;Return to rom
     DB 0F,0F,0F,0F,0F ;1BEC-1BF0

;VERIFY / handler
VF/H CALL VFER   ;1BF1-1BF3
     DB 0F,0F,0F,0F,0F ;1BF4-1BF8
     PUSH HL     ;1BF9
     DB 0F,0F    ;1BFA-1BFB
     DB 0F,0F,0F,0F,0F ;1BFC-1CF0
     CALL LCAT   ;Load the catalog
VFCNT LD DE,TCAT   ;Point at name to find
     LD HL,CTFL   ;Point at start of CAT entry names
     CALL MTCH    ;Go & find a match
     JP NC,ER_S   ;Error S if file not found
     LD DE,TCAT   ;Move main cat entry to TCAT area
     LD BC,0014
     LDIR
     LD A,(TSID)  ;Set SID# sv to file beg side#
     LD (SID#),A
     LD A,(TTRK)  ;Set TRK# sv to file beg track#
     LD (TRK#),A
     CALL FND#    ;Set ddriv/side port & DSEL to match sid#
     CALL RSR2    ;Restore drive
     CALL SEKT    ;Seek track (TRK#)
     JP NZ,ER_T   ;Disk I/O error if not found
VFLP CALL VFCY    ;Verify a cylinder
     RET NC       ;Ret w/NC to signal error if needed
     CALL NEXCY   ;Advance to next cylinder
     LD HL,TCYL   ;Point at file's # of cylinder size
     DEC (HL)     ;Dec cyl size
     JR NZ,VFLP   ;Loop till all cylinders are checked
     SCF          ;Signal no errors
     RET

;NEXT command routine
NEXT LD A,(flgs)  ;Syntax time?
     RLA
     JR NC,DSYN   ;Skip in syntax time
```

```
     LD A,(FORF)  ;Get FOR / flag
     INC A        ;Test it
     JR NZ,ER_1   ;Error 1 if FOR / not active
     LD SP,(vars) ;Point at variables
     INC SP       ;
     POP HL
     OR L
     JR NZ,ERY2
     OR H
     JR NZ,ERZ2
     POP DE
     INC DE
     PUSH DE
     LD HL,(LPCN)
     SBC HL,DE
     JR C,DONE3
     LD SP,NXC'
     POP HL
     LD (chad),HL
     POP HL
     LD (ppc_),HL
     POP HL
     LD (nxln),HL
     DEC SP
     POP AF
     LD (sbpc),A
DON3 LD SP,(ersp)
     DEC SP
     DEC SP
     RST 18H
DONE3 LD (FORF),A
     JR DON3

;Error 1
ER_1 LD L,00
     JP ER_L

ERY2 JP ER_Y

ERZ2 JP ER_Z

DSYN LD SP,(ersp) ;Get error stack pointer address
     DEC SP       ;Set stack pointer two words deep
     DEC SP
     DEC SP
     DEC SP
     INC HL       ;Point at next character
     LD A,(HL)    ;Get next character
     CP 3A        ;Character ":"?
```

```
     JR Z,ENOK    ;OK if it is
     CP 0D        ;Character CR?
     JP NZ,ER_C   ;Error C if not
ENOK JP SYNF

;Erase command handler
H/ER ORG 1CB2     ;4 bytes before ERAS ORG
     CALL VFER    ;Get file name & ret in syntax time
     PUSH HL      ;HL will be pointed at DONE to handle rom ret

;ERASE /fn command routine
ERAS ORG 1CB6     ;Erase command address is documentated
     CALL LCAT    ;Load catalog
     LD DE,TCAT   ;Point at filename & type to find
     LD HL,CTFL   ;Point at start of catalog in buffer
     CALL MTCH    ;Look for a match
     JP NC,ER_S   ;Error S if no match found
;Now, HL points to matching CAT file entry beginning
     LD A,(DSDS)  ;Get # of sides on this disk
     LD (SSDS),A  ;Save as # of sides flag
     LD (EADR),HL ;EADR points to current file to erase in CAT
     LD DE,0013   ;Point at file's size in # of cyls
     ADD HL,DE
     LD A,(HL)    ;Get # of cyls reserved for file
     LD (SIZE),A  ;SIZE is # of cylinders to reclaim
     DEC HL       ;Point to files starting side#
     LD A,(HL)    ;Get starting side#
     LD (DESD),A  ;Now is dest starting side#
     LD (SID#),A  ;And current side#
     CALL FND#    ;Form DSEL from this side#
     LD (DSL2),A  ;Save this DSEL
     DEC HL       ;Point at files starting track#
     LD A,(HL)    ;Get files starting track#
     LD (DTK#),A  ;Now is dest starting track#
NXPG LD DE,CBUF   ;Point at start of erase buffer (3400-34E0H)
     LD (FLAD),DE ;Set erase buff file pointer to start
     LD HL,(EADR) ;Get location within cat of dest CAT entry
     LD BC,00DC   ;DCh bytes to move means 11 CAT entries
     LDIR         ;Move these to erase buffer
ERSLOOP LD HL,(FLAD) ;Get erase buffer file pointer
     LD A,L       ;LSB of address to A
     CP C8        ;11th file?
     JR Z,AGAIN   ;Jump to update & load another 10 if so
     PUSH HL      ;Save erase buffer pointer
     LD DE,TCAT   ;Move next CAT entry to TCAT
     LD BC,0014
     ADD HL,BC
     LD A,(HL)    ;Get 1rst byte of filename
     LDIR
```

```
     POP HL       ;Get erase buffer pointer
     CP 80        ;This file end of CAT?
     JR Z,EEND    ;Jump if end reached
     EX DE,HL     ;DE=erase buffer file pointer
     LD HL,(TTRK) ;Get next file's beg track#
     LD A,L
     LD (STK#),A  ;Now source track#
     LD A,H       ;Get next file's beg side#
     LD (SRSD),A  ;Now src side#
     LD (SID#),A  ;And current side#
     CALL FND#    ;Derive DSEL fron this side#
     LD (DSL'),A  ;Save src DSEL
     LD HL,(DTK#) ;Get dest track#
     LD A,(DESD)  ;And dest side#
     LD H,A
     LD (TTRK),HL ;Now temp file's track/side#
     LD HL,TCAT   ;Overwrite last CAT entry with this one
     LD BC,0014
     LDIR
     LD (FLAD),DE ;Save erase buffer file pointer
     CALL DOWN    ;Move the actual file down
     JR C,ERSLOOP ;Loop for next file if no error
     JP ER_T      ;Ret w/report T. Disk I/O error

;Here after end of CAT found
EEND LD (HL),A    ;Mark end of CATalog w/80h
     CALL LCAT    ;Load the CAT
     LD HL,(NXCA) ;Get next available CAT entry address
     LD DE,FFEC   ;Now 20d bytes less
     ADD HL,DE
     LD (NXCA),HL ;Store updated next avail CAT entry address
     LD HL,(DTK#) ;Get newly vacated track#
     LD A,(DESD)  ;And side#
     LD H,A
     LD (NXCY),HL ;And store as next avail cyl address
     LD HL,(FRCY) ;Get total free cyls left on disk
     LD DE,(SIZE) ;Get # of cylinders reclaimed
     LD D,00      ;No more than 255
     ADD HL,DE    ;Add newly reclaimed cyls to avail count
     LD (FRCY),HL ;Store new count
U&SC LD HL,CBUF   ;Point at erase buffer
     LD DE,(EADR) ;Point at CAT area to insert it
     LD BC,00DC   ;11 file entries to move
     LDIR         ;Move revised entries to actual CATalog
     CALL SCAT    ;Save updated CATalog
     JP SIZ0'     ;Return via set-size-to-zero routine

AGAIN CALL LCAT   ;Load CATalog from disk
     LD HL,CBUF   ;Point at erase buffer
```

```
     LD DE,(EADR) ;Point at portion of CATalog to be revised
     LD BC,00DC   ;11 entries to be moved
     LDIR         ;Move them
     LD HL,FFEC   ;But point at 11th entry
     ADD HL,DE
     LD (EADR),HL ;This entry is now current CAT addr pointer
     CALL SCAT    ;Save revised CATalog
     JP NXPG      ;Jump to handle next page of 10 entries

;Here to get filename & ret to Basic rom in syntax time
VFER RST 10H      ;Advance to next Basic character
     POP DE       ;Save calling address
     CALL RSTK    ;Reset SP
     PUSH DE      ;Ret address back to stack
     CALL NXEX    ;Evaluate next expression
     JP C,ER_C    ;Error C if numeric expression
     CALL GTST
     POP DE
     CALL SYRT    ;Ret to Basic rom in syntax time
     PUSH DE
     LD HL,DONE
     RET

DOWN CALL SLSR    ;Select src cyl address
     CALL SEKT    ;Seek track (TRK#)
     RET NZ       ;Ret w/NC if error
     CALL L5KB    ;Load 5K bytes
     RET NC       ;Ret w/NC if error
     CALL NXCX    ;Find next cylinder
     CALL SVSC    ;Save src cyl address
     LD A,(TRK#)  ;Get current track#
     LD (STK#),A  ;Store src cyl track#
     CALL SLDE    ;Select dest cyl address
     CALL SEKT    ;Seek dest track#
     RET NZ       ;Ret w/NC if error
     CALL S5KB    ;Save 5K bytes
     RET NC       ;Ret w/NC if error
     CALL NXCX    ;Find next cylinder
     CALL SVDS    ;Save dest cyl address
     LD A,(TRK#)  ;Get current track#
     LD (DTK#),A  ;Store dest cyl track#
     LD HL,TCYL   ;Point to file cyl length
     DEC (HL)     ;Dec count
     JR NZ,DOWN   ;Loop till all cyls in file moved
     SCF          ;Signal no errors
     RET

;Get (HL) from bank FF into reg L
GTHL PUSH AF      ;Save all registers
```

```
     PUSH BC
     PUSH DE
     PUSH HL
     LD HL,mem5    ;Save 7 of Basic's bytes
     LD DE,STOR    ;in BRAM
     LD BC,0007
     PUSH BC
     LDIR
     POP BC        ;Count=7 again
     PUSH BC
     DEC HL        ;Now move SAFE's code into mem5 ram
     EX DE,HL
     LD HL,CDEND+2
     LDDR
     POP BC
     POP HL
     CALL mem5     ;Call SAFE code in HOME ram
     PUSH HL       ;Now move Basic's bytes back
     INC DE
     LD HL,STOR
     LDIR
     POP HL        ;Restore all registers
     POP DE
     POP BC
     POP AF
     RET

;This is the 7 bytes moved to run from ram
;Routine will return (HL) in L from other bank
RMCD LD A,(0008)   ;Turn off Bbank
     LD L,(HL)     ;Get byte pointed to
CDEND JP B_ON      ;Turn B bank back on & return

FNDM PUSH HL
     LD A,(SV/L)   ;Get SAVE/MERGE/LOAD flag
     AND A         ;Test flag
     JR NZ,LDMT    ;Jump if MERGE or LOAD
;Here if match found & SAVE in progress
     LD BC,0013
     ADD HL,BC
     CALL OVWR
     LD A,(TCYL)
     LD C,A
     LD A,(HL)
     CP C
     JP C,ER_V
     LD (TCYL),A
     LD A,FF       ;Signal copy-over w/flag
     LD (COVR),A
```

```
     DEC HL
     LD D,(HL)
     DEC HL
     LD E,(HL)
     LD (TTRK),DE
     POP HL
     LD (FLAD),HL
SL9X JP S/L9

;Here if file found & MERGE or LOAD in progress
LDMT LD DE,(TLEN)
     PUSH DE
     LD DE,(TBEG)
     PUSH DE
     LD DE,TCAT
     LD BC,0014
     LDIR
     LD A,(TTYP)
     CP 03
     POP DE
     JR Z,CODL
     POP DE
     POP HL
     CP 04
     JR Z,SL9X
     AND A
     JP Z,PGLM
     CP 01
     JR Z,NARR
     CP 02
NARR JP Z,DTLM    ;Jump if DATA load or merge
     JP VRLM

CODL LD A,(CODF)
     RRA
     JR C,DFBG
     LD (TBEG),DE
DFBG POP DE
     POP HL
     RRA
     JR C,S/L.9
     LD HL,(TLEN)
     SBC HL,DE
     JR C,S/L.9
     LD (TLEN),DE
S/L.9 JR S/L9

;Sinclair rom entry point w/o interrupts enabled
ORG 1E8F
```

```
SRNI LD A,(0008)  ;Next instruction will be @1E92 in Spec rom

;Here to find pixel address
PXAD LD B,A
     AND A
     RRA
     SCF
     RRA
     AND A
     RRA
     XOR B
     AND F8
     XOR B
     LD H,A
     LD A,C
     RLCA
     RLCA
     RLCA
     XOR B
     AND C7
     XOR B
     RLCA
     RLCA
     LD L,A
     LD A,C
     AND 07
     RET

;Save/Load/Merge command routine
ORG 1EAD
SL@N CALL LCAT    ;Load CATalog
S/L4 XOR A        ;Reset copy-over flag
     LD (COVR),A
S/L5 LD HL,(TLEN) ;Get file length
     LD DE,1400   ;Each cyl=5K
     LD BC,0000   ;BC=cyl counter
CYLP AND A        ;Clear carry
     SBC HL,DE    ;HL=HL-5K
     INC BC       ;Inc cyl count
     JR Z,DONE!   ;Done if result=00
     JR NC,CYLP   ;Not done if NZ & NC
DONE! LD A,C
     LD (TCYL),A
S/L6 LD DE,TCAT
     LD HL,CTFL
     CALL MTCH
     JP C,FNDM
     LD A,(SV/L)
     AND A
```

```
     JP NZ,ER_S
     LD A,(TCYL)
     LD C,A
     LD B,00
     LD HL,(FRCY)
     AND A
     SBC HL,BC
     JR C,ER_U
S/L7 LD HL,(NXCA)
     LD DE,33E0
     EX DE,HL
     AND A
     SBC HL,DE
     JR C,ER_U
S/L8 LD HL,(NXCY)
     LD (TTRK),HL
S/L9 LD A,(TTRK)
     LD (TRK#),A
     LD A,(TSID)
     LD (SID#),A
     CALL FND#
     CALL RSR2
     CALL SEKT
     JR NZ,SL12
SL10 LD HL,(TBEG)
     JR OVERT     ;Jump over Timex rom entry point

;Timex rom entry point w/o interrupts
ORG 1F19
TRNI LD A,(0008)

OVERT LD A,(TTYP)
     AND A
     JR NZ,NOPG
     LD HL,(prog)
NOPG LD DE,(TLEN)
     PUSH HL
     DEC HL
     ADD HL,DE
     POP HL
ERRB2 JP C,ER_B
     LD A,D
     OR E
     SCF
     RET Z
     PUSH HL
     LD BC,0070
     AND A
     SBC HL,BC
```

```
     POP HL
     JR C,ERRB2
     LD A,(SV/L)
     AND A
     JR Z,SAVE
     CALL LDHD
     JR NC,SL12
     RET

SAVE CALL SVHD
SL11 JR C,SL18
SL12 CALL RTRY
     JR S/L9

;Error U
ER_U LD A,02      ;New error#2
     JP NERR      ;Stop w/new error 2

SL18 LD DE,TCAT
     LD HL,(NXCA)
     LD BC,0014
     LD A,(COVR)
     AND A
     JR Z,MVCT
     LD HL,(FLAD)
MVCT EX DE,HL
     LDIR
     AND A
     JR NZ,CPOV
     LD A,80
     LD (DE),A
     LD (NXCA),DE
     CALL NEXCY
     LD A,(SID#)
     LD H,A
     CALL REDY    ;Will ret here w/NC if ready
     IN A,(9F)
     LD L,A
     LD (NXCY),HL
     LD A,(TCYL)
     LD C,A
     LD B,00
     LD HL,(FRCY)
     SBC HL,BC
     LD (FRCY),HL
CPOV JP SCAT

MDNA AND A
     JP Z,ER_F
```

```
TRUN CP 11
     JR C,LESS17
     LD A,10
LESS17 LD C,A
     LD B,00
     LD DE,DNAM
     LDIR
FILL LD A,E
     AND 0F
     RET Z
     LD A,20
     LD (DE),A
     INC DE
     JR FILL

;Test BREAK key & return w/BREAK report if pressed
TBRK LD A,7F      ;Scan keyrow containing break key
     IN A,(FE)
     RRA
     RET C        ;Return if not pressed
     LD A,FE      ;But both space & shift must be pressed
     IN A,(FE)
     RRA
     RET C        ;Return unless both are pressed
     LD A,(SIZE)  ;Are we in the middle of an ERASE?
     AND A
     RET NZ       ;No breaks allowed if we are
     IN A,(8F)    ;Get controller status
     RRA          ;Busy bit to carry
     JR NC,ER_D   ;Skip if not busy
     LD A,D0      ;D0 will stop controller anytime
     OUT (8F),A   ;Force a stop

;Error D; BREAK key pressed
ER_D LD L,0C      ;Signify error D
     JP ER_L      ;BREAK key pressed

;Ascii screen copy routine
ACOP LD BC,0000   ;Start at col 0, line 0
CPLP PUSH BC      ;Save col & line
     CALL SC$C    ;Get char code at this loc
CONT: INC C        ;Valid code?
     DEC C
     JR NZ,PRCD   ;Print it if it is
     LD A,20      ;Or print space if not
PRCD CALL PRTA    ;Print the char
     POP BC       ;Restore col/line pointer
     INC B        ;Next col
     LD A,1F      ;32 chars per line
```

```
CP B          ;Was that #32?
JR NC,CPLP    ;Loop for next if not
LD A,0D       ;It was, so send CR
CALL PRTA
LD A,(5B1C)   ;(5B1C)=code to follow CR (00 or 0A)
CALL PRTA
LD B,00       ;Reset col pointer
INC C         ;Next line
LD A,17       ;24 lines per screen
CP C          ;Was that #24?
JR NC,CPLP    ;Loop for next line if not
RET           ;Else ret

ORG 1FFF      ;Last byte of eprom contains SAFE 2 version
DB 65         ;65H is current version
```